1. KEŠ MEMORIJA 4

1.1. OPŠTE NAPOMENE 4
1.1.1. Tehnike preslikavanja 5
1.1.1.1. Direktno preslikavanje 5
1.1.1.2. Asocijativno preslikavanje 6
1.1.1.3. Set-asocijativno preslikavanje 8
1.1.2. Dimenzija bloka 13
1.1.3. Zamena blokova keš memorije 13
1.1.4. Ažuriranje operativne memorije 14
1.1.5. Neka razmatranja u vezi realizacije keš memorije 15
1.2. REALIZACIJA KEŠ MEMORIJA 16
1.2.1. keš memorija sa Direktnim preslikavanjem 16
1.2.1.1. PROCESOR CPU 17
1.2.1.1.1. Operaciona jedinica 19
1.2.1.1.1.1. Blok keš interfejs 19
1.2.1.1.1.2. Blok generator operacija 22
1.2.1.1.2. Upravljačka jedinica 23
1.2.1.1.2.1. Dijagram toka generisanja operacija 23
1.2.1.1.2.2. Algoritam generisanja upravljačkih signala 25
1.2.1.1.2.3. Vremenski oblici signala 27
1.2.1.1.2.4. Struktura upravljačke jedinice 32
1.2.1.1.2.4.1. Blok generisanje nove vrednosti brojača koraka 33
1.2.1.1.2.4.2. Blok brojač koraka 33
1.2.1.1.2.4.3. Blok dekoder koraka 34
1.2.1.1.2.4.4. Blok generisanje upravljačkih signala 34
1.2.1.1.2.4.4.1. Upravljački signali operacione jedinice 34
1.2.1.1.2.4.4.1.1. Blok keš interfejs 34
1.2.1.1.2.4.4.1.2. Blok generisanje operacija 35
1.2.1.1.2.4.4.2. Upravljački signali upravljačke jedinice 35
1.2.1.2. MEMORIJA MEM 36
1.2.1.3. KEŠ MEMORIJA KEŠ 36
1.2.1.3.1. Operaciona jedinica 37
1.2.1.3.1.1. Blok cpu interfejs 38
1.2.1.3.1.2. Blok indikatori 41
1.2.1.3.1.3. Blok brojači 43
1.2.1.3.1.4. Blok tag memorija 45
1.2.1.3.1.5. Blok data memorija 46
1.2.1.3.1.6. Blok mem interfejs 48
1.2.1.3.2. Upravljačka jedinica 49
1.2.1.3.2.1. Dijagram toka operacija 49
1.2.1.3.2.2. Algoritam generisanja upravljačkih signala 53
1.2.1.3.2.3. Vremenski oblici signala 56
1.2.1.3.2.3.1. Operacija čitanja 56
1.2.1.3.2.3.2. Operacija upisa 60
1.2.1.3.2.3.3. Operacija selektivnog vraćanja 64
1.2.1.3.2.3.4. Operacija kompletnog vraćanja 67
1.2.1.3.2.4. Struktura upravljačke jedinice 69
1.2.1.3.2.4.1. Blok generisanje nove vrednosti brojača koraka 70
1.2.1.3.2.4.2. Blok brojač koraka 71
1.2.1.3.2.4.3. Blok dekoder koraka 71
1.2.1.3.2.4.4. Blok generisanje upravljačkih signala 71
1.2.1.3.2.4.4.1. Upravljački signali operacione jedinice 72
1.2.1.3.2.4.4.1.1. Blok cpu interfejs 72
1.2.1.3.2.4.4.1.2. Blok indikatori 72
1.2.1.3.2.4.4.1.3. Blok brojači 73
1.2.1.3.2.4.4.1.4. Blok tag memorija 73
1.2.1.3.2.4.4.1.5. Blok data memorija 73
1.2.1.3.2.4.4.1.6. Blok mem interfejs 74
1.2.1.3.2.4.4.2. Upravljački signali upravljačke jedinice 74
1.2.2. keš memorija sa Asocijativnim preslikavanjem 75
1.2.2.1. PROCESOR CPU 76
1.2.2.1.1. Operaciona jedinica 77
1.2.2.1.1.1. Blok keš interfejs 77
1.2.2.1.1.2. Blok generator operacija 79
1.2.2.1.2. Upravljačka jedinica 81

1.2.2.1.2.1. Dijagram toka generisanja operacija ... 81
1.2.2.1.2.2. Algoritam generisanja upravljačkih signala ... 83
1.2.2.1.2.3. Vremenski oblici signala ... 84
1.2.2.1.2.4. Struktura upravljačke jedinice ... 86
1.2.2.1.2.4.1. Blok generisanje nove vrednosti brojača koraka ... 86
1.2.2.1.2.4.2. Blok brojač koraka ... 86
1.2.2.1.2.4.3. Blok dekoder koraka ... 87
1.2.2.1.2.4.4. Blok generisanje upravljačkih signala ... 87
1.2.2.1.2.4.4.1. Upravljački signali operacione jedinice ... 87
1.2.2.1.2.4.4.1.1. Blok keš interfejs ... 87
1.2.2.1.2.4.4.1.2. Blok generisanje operacija ... 88
1.2.2.1.2.4.4.2. Upravljački signali upravljačke jedinice ... 88
1.2.2.2. MEMORIJA MEM ... 89
1.2.2.3. KEŠ MEMORIJA KEŠ ... 89
1.2.2.3.1. Operaciona jedinica ... 90
1.2.2.3.1.1. Blok cpu interfejs ... 91
1.2.2.3.1.1. Indikatori ... 93
1.2.2.3.1.2. Blok brojači ... 94
1.2.2.3.1.3. Blok tag memorija ... 96
1.2.2.3.1.4. Blok data memorija ... 97
1.2.2.3.1.5. Blok lru kola ... 99
1.2.2.3.1.5.1. Algoritam zamene ... 99
1.2.2.3.1.5.2. Strukturna šema ... 100
1.2.2.3.1.6. Blok mem interfejs ... 103
1.2.2.3.2. Upravljačka jedinica ... 104
1.2.2.3.2.1. Dijagram toka operacija ... 104
1.2.2.3.2.2. Algoritam generisanja upravljačkih signala ... 106
1.2.2.3.2.3. Vremenski oblici signala ... 109
1.2.2.3.2.3.1. Operacija čitanja ... 109
1.2.2.3.2.3.2. Operacija upisa ... 111
1.2.2.3.2.4. Struktura upravljačke jedinice ... 113
1.2.2.3.2.4.1. Blok generisanje nove vrednosti brojača koraka ... 114
1.2.2.3.2.4.2. Blok brojač koraka ... 115
1.2.2.3.2.4.3. Blok dekoder koraka ... 115
1.2.2.3.2.4.4. Blok generisanje upravljačkih signala ... 115
1.2.2.3.2.4.4.1. Upravljački signali operacione jedinice ... 116
1.2.2.3.2.4.4.1.1. Blok cpu interfejs ... 116
1.2.2.3.2.4.4.1.2. Blok indikatori ... 116
1.2.2.3.2.4.4.1.3. Blok brojači ... 116
1.2.2.3.2.4.4.1.4. Blok tag memorija ... 117
1.2.2.3.2.4.4.1.5. Blok data memorija ... 117
1.2.2.3.2.4.4.1.6. Blok lru kola ... 117
1.2.2.3.2.4.4.1.7. Blok mem interfejs ... 118
1.2.2.3.2.4.4.2. Upravljački signali upravljačke jedinice. ... 118
1.2.3. KEŠ memorija sa Set-asocijativnim preslikavanjem ... 119
1.2.3.1. PROCESOR CPU ... 120
1.2.3.1.1. Operaciona jedinica ... 123
1.2.3.1.1.1. Blok kešinterfejs ... 123
1.2.3.1.1.2. Blok operacije ... 125
1.2.3.1.2. Upravljačka jedinica ... 126
1.2.3.1.2.1. Dijagram toka operacija ... 127
1.2.3.1.2.2. Sekvenca upravljačkih signala po koracima ... 129
1.2.3.1.2.3. Vremenski oblici signala ... 130
1.2.3.1.2.4. Struktura upravljačke jedinice ... 131
1.2.3.1.2.4.1. Blok generisanje nove vrednosti brojača koraka ... 132
1.2.3.1.2.4.2. Blok brojač koraka ... 132
1.2.3.1.2.4.3. Blok dekoder koraka ... 133
1.2.3.1.2.4.4. Blok generisanje upravljačkih signala ... 133
1.2.3.1.2.4.4.1. Upravljački signali operacione jedinice ... 133
1.2.3.1.2.4.4.2. Upravljački signali upravljačke jedinice ... 134
1.2.3.2. MEMORIJA MEM ... 134
1.2.3.3. KEŠ MEMORIJA KEŠ ... 135
1.2.3.3.1. Operaciona jedinica ... 135
1.2.3.3.1.1. Blok cpuinterfejs ... 136
1.2.3.3.1.2. Blok indikatori ... 139
1.2.3.3.1.3. Blok brojači ... 142
1.2.3.3.1.4. Blok tagmem ... 144

1.2.3.3.1.5. Blok datamem 146
1.2.3.3.1.6. Blok fifokola 150
1.2.3.3.1.7. Blok meminterfejs 151
1.2.3.3.2. Upravljačka jedinica 153
1.2.3.3.2.1. Dijagrami toka operacija 153
1.2.3.3.2.2. Sekvenca upravljačkih signala po koracima 158
1.2.3.3.2.3. Struktura upravljačke jedinice 164
1.2.3.3.2.3.1. Blok generisanje nove vrednosti brojača koraka 166
1.2.3.3.2.3.2. Blok brojač koraka 166
1.2.3.3.2.3.3. Blok dekoder koraka 167
1.2.3.3.2.3.4. Blok generisanje upravljačkih signala 167
1.2.3.3.2.3.4.1. Upravljački signali operacione jedinice 167
1.2.3.3.2.3.4.2. Upravljački signali upravljačke jedinice 169

1. KEŠ MEMORIJA

U ovoj glavi se najpre daju neke opšte napomene o keš memorijama, zatim se prikazuju realizacije keš memorija sa direktnim, asocijativnim i set-asocijativnim preslikavanjem i na kraju daju informacije neophodne za korišćenje simulatora i zadaci.

1.1. OPŠTE NAPOMENE

U ovom poglavlju se najpre prikazuje kako se korišćenjem keš memorije ubrzava pristup memorijskim lokacijama. Zatim se razmatraju osnovna pitanja vezana za realizaciju keš memorije, i to tehnika preslikavanja, veličina bloka, zamena blokova keš memorije i ažuriranje sadržaja operativne memorije. Na kraju se daju neka specifična rešenja čijim se korišćenjem poboljšava rad keš memorija.

Uvođenje keš memorije je bazirano na pretpostavci da ako Vreme izvršavanja instrukcija, a samim tim i programa, direktno zavisi od vremena pristupa operativnoj memoriji. Pristup operativnoj memoriji se može ubrzati korišćenjem keš memorije koja se ubacuje između procesora i operativne memorije. Keš memorija se realizuje tako da je njeno vreme pristupa daleko manje od vremena pristupa operativne memorije. S obzirom da je cena po bitu keš memorije značajno veća od cene po bitu operativne memorije, kapacitet keš memorije je daleko manji od kapaciteta operativne memorije. U keš memoriji se čuvaju sadržaji onih lokacija operativne memorije kojima se procesor najčešće obraća. Algoritam izvršavanja instrukcija u računarima sa keš memorijom predviđa da se pri svakom obraćanju procesora operativnoj memoriji najpre izvrši provera da li je dati sadržaj u keš memoriji. Ako jeste, čitanje ili upis se vrši sa keš memorijom. Ako nije, sadržaj date lokacije se, najpre, prebaci iz operativne u keš memoriju pa se, potom, čitanje ili upis realizuje sa keš memorijom. Kompletan rad sa keš memorijom se realizuje hardverski.

je iz programa pristupano nekim lokacijama operativne memorije, postoji velika verovatnoća da će im se uskoro ponovo pristupiti. Ova pretpostavka je utemeljena na uočenom prostornom i vremenskom lokalitetu programa. Prostorni lokalitet znači da su adrese koje će se uskoro generisati u neposrednoj blizini adresa koje su upravo generisane. Ovakav način generisanja adresa se javlja kod generisanja adresa elemenata nizova, čiji su elementi u operativnoj memoriji smešteni jedan za drugim, ili kod instrukcija, koje se izvršavaju sekvencijalno. Vremenski lokalitet znači da se određene adrese, koje prostorno u operativnoj memoriji ne moraju da budu bliske, više puta generišu u određenom vremenskom intervalu. Ovakav način generisanja adresa se javlja kod generisanja adresa skalarnih promenljivih.

Keš memorije je moguće realizovati na više načina. Razlike postoje u tehnici preslikavanja, dimenziji bloka, zameni blokova keš memorije i ažuriranju operativne memorije. **Tehnika preslikavanja** određuje način vođenja evidencije o tome koji se delovi operativne memorije nalaze u pojedinim delovima keš memorije. **Dimenzija bloka** određuje broj reči operativne memorije na nivou kojih se realizuje preslikavanje. **Zamena blokova keš memorije** određuje ulaz keš memorije u koji treba smestiti blok koji se dovlači iz operativne memorije u slučaju kada je keš memorija popunjena. **Ažuriranje operativne memorije** određuje kako se kod operacije upisa obezbeđuje ne samo promena sadržaja u keš, već i u operativnoj memoriji.

1.1.1. TEHNIKE PRESLIKAVANJA

Najčešće se koriste sledeće tri tehnike preslikavanja:

- direktno,
- asocijativno i
- set-asocijativno.

1.1.1.1. DIREKTNO PRESLIKAVANJE

Kod tehnike direktnog preslikavanja keš memorija se sastoji iz sledećih delova (slika 1):

- memorije DATA i
- memorije TAG.

U memoriju DATA se smeštaju blokovi preneti iz operativne u keš memoriju. U memoriju TAG se smeštaju brojevi grupa, koji se nazivaju TAG-ovi, za blokove prenete iz operativne u keš memoriju. Za realizaciju memorija DATA i TAG koristi se RAM memorija. Ukoliko u memoriju DATA može da se smesti 2^k blokova operativne memorije kaže se da keš memorija ima 2^k ulaza. Za svaki ulaz memorije DATA postoji odgovarajući ulaz memorije TAG.

U slučaju keš memorije sa 2^k ulaza realizovane u tehnici direktnog preslikavanja zamišlja se kao da je operativna memorija kapaciteta 2^{m+k} blokova podeljena na 2^m grupa pri čemu veličina grupe odgovara veličini memorije DATA i iznosi 2^k blokova. Ukoliko se uzme da je dimenzija bloka 2^l reči, tada generisana adresa ima sledeću strukturu:

- najnižih l bita određuju adresu reči unutar bloka,
- srednjih k određuju broj bloka unutar grupe i
- najviših m bita određuju broj grupe u operativnoj memoriji kojoj pripada blok.

U ovoj tehnici svaki blok operativne memorije ima fiksno definisan ulaz u keš memoriji u koji se smešta prilikom dovlačenja iz operativne u keš memoriju. Tako se i-ti blok iz bilo koje od 2^m grupa operativne memorije prilikom dovlačenja može da smesti jedino u i-ti ulaz memorije DATA. Pošto se u i-tom ulazu memorije DATA može da nađe i-ti blok iz bilo koje od 2^m grupa operativne memorije, u i-tom ulazu memorije TAG se čuva broj grupe kojoj pripada dati blok. S obzirom da u operativnoj memoriji ima 2^m grupa, širina memorijske reči memorije TAG je m bita.

Pri generisanju zahteva za čitanje od strane procesora srednjih k bita generisane adrese određuju ulaz u memorijama TAG i DATA. Sadržaj adresirane lokacije memorije TAG poredi se sa najviših m bita generisane adrese da bi se utvrdilo da li postoji saglasnost ili ne. Ukoliko postoji saglasnost, sa srednjih k i najnižih l bita generisane adrese adresira se reč memorije DATA i obavlja čitanje. Ako saglasnost ne postoji, najpre se ceo blok iz ulaza memorije DATA koji je određen sa srednjih k bita generisane adrese prebacuje u operativnu memoriju. Broj grupe operativne memorije u koju se blok upisuje određen je sa m bita

očitanih iz memorije TAG i to sa ulaza određenog sa srednjih *k* bita generisane adrese. Broj bloka unutar date grupe operativne memorije određen je sa srednjih *k* bita generisane adrese. Potom se ceo blok u kome se nalazi adresirani sadržaj dovlači iz operativne memorije i smešta u ulaz memorije DATA koji je određen sa srednjih *k* bita generisane adrese. Takođe se broj grupe kojoj dati blok pripada, određen sa najviših *m* bita generisane adrese, upisuje u memoriju TAG i to u ulaz određen sa srednjih *k* bita generisane adrese. Potom se na već opisani način vrši provera da li postoji saglasnost, utvrđuje da postoji, i realizuje čitanje.

Pri generisanju zahteva za upis od strane procesora na isti način se ispituje saglasnost sa sadržajem keš memorije kao u slučaju operacije čitanja. Ukoliko postoji saglasnost, sa srednjih *k* i najnižih *l* bita generisane adrese adresira se reč memorije DATA i nakon toga vrši se upis. Ako saglasnost ne postoji, na identičan način kao i za operaciju čitanja se najpre blok iz ulaza keš memorije određenog sa srednjih *k* bita generisane adrese prebacuje u operativnu memoriju, a zatim u isti ulaz keš memorije dovlači novi blok iz operativne memorije. Potom se, ponovo, na već opisani način vrši provera da li postoji saglasnost, utvrđuje da postoji, i realizuje upis.

Opisani mehanizam čitanja iz i upisa u keš memoriju je osnovni mehanizam rada sa keš memorijom. U praktičnim realizacijama postoji više varijanti ovog osnovnog mehanizma.

Prednost tehnike direktnog preslikavanja je da se za realizaciju keš memorije koriste standardne RAM memorije i da je algoritam zamene trivijalan. Nedostatak je da se u isti blok keš memorije preslikava 2^m blokova operativne memorije, tako da dolazi do zamene starog bloka novim i onda kada nije popunjena cela keš memorija.

1.1.1.2. ASOCIJATIVNO PRESLIKAVANJE

Kod tehnike asocijativnog preslikavanja keš memorija se sastoji iz sledećih delova (slika 2):

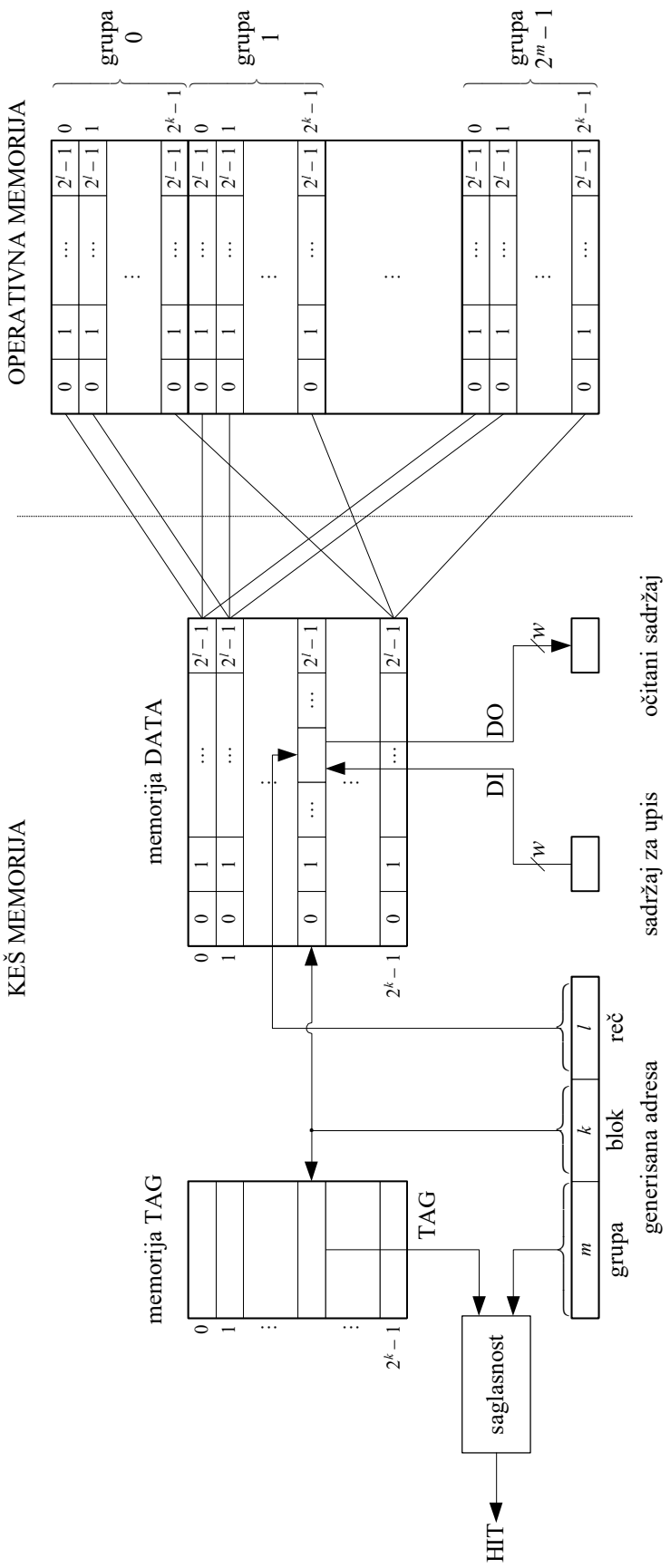
- memorije DATA i
- memorije TAG.

U memoriju DATA se smeštaju blokovi preneti iz operativne u keš memoriju. U memoriju TAG se smeštaju brojevi blokova, koji se nazivaju TAG-ovi, za blokve prenete iz operativne u keš memoriju. Za realizaciju memorija DATA i TAG koriste se RAM i asocijativna memorija, respektivno. Ukoliko u memoriju DATA može da se smesti 2^n blokova kaže se da keš memorija ima 2^n ulaza. Za svaki ulaz memorije DATA postoji odgovarajući ulaz memorije TAG.

U slučaju keš memorije realizovane u tehnici asocijativnog preslikavanja zamišlja se kao da je operativna memorija podeljena na 2^h blokova. Ukoliko se uzme da je dimenzija bloka 2^l reči tada generisana adresa ima sledeću strukturu:

- nižih *l* bita određuju adresu reči unutar bloka i
- viših *h* bita određuju broj bloka u operativnoj memoriji.

U ovoj tehnici preslikavanja bilo koji blok operativne memorije može da se smesti u bilo koji ulaz keš memorije prilikom dovlačenja iz operativne u keš memoriju. Pošto se u *i*-tom ulazu memorije DATA može da nađe bilo koji od 2^h blokova operativne memorije, u *i*-tom ulazu memorije TAG se čuva broj bloka operativne memorije kome pripada dati blok. S obzirom da u operativnoj memoriji ima 2^h blokova, širina memorijske reči memorije TAG je *h* bita.



Slika 1 Keš memorija sa direktnim preslikavanjem

Pri generisanju zahteva za čitanje od strane procesora viših *h* bita generisane adrese vodi se na ulaz memorije TAG da bi se, njihovim istovremenim upoređivanjem sa sadržajima svih 2^n ulaza memorije TAG, utvrdilo da li postoji saglasnost sa sadržajem nekog ulaza ili ne. Za svaki od 2^n ulaza memorije TAG postoji poseban signal [ ] do [ ] koji svojom aktivnom vrednošću određuje da je na datom ulazu otkrivena saglasnost. Utvrđena saglasnost na *i*-tom ulazu memorije TAG znači da se adresirana reč nalazi u bloku u *i*-tom ulazu memorije DATA. Binarna vrednost broja ulaza u kome je otkrivena saglasnost je određena sa *n* bita sa izlaza kodera na osnovu signala [ ] do [ ]. Signal saglasnosti **HIT**, koji se dobija sa izlaza kodera W, ima aktivnu vrednost ukoliko jedan od signala [ ] do [ ] ima aktivnu vrednost. Ukoliko postoji saglasnost, sa *n* bita sa izlaza kodera i *l* bita generisane adrese adresira se reč memorije DATA i obavlja čitanje. Ako saglasnost ne postoji, na osnovu odabranog algoritma zamene, određuje se ulaz keš memorije za zamenu. Najpre se ceo blok iz odabranog ulaza memorije DATA prebacuje u operativnu memoriju. Broj bloka operativne memorije u koji se blok upisuje određen je sa *h* bita očitanih iz memorije TAG i to sa ulaza određenog za zamenu. Potom se ceo blok u kome se nalazi adresirani sadržaj dovlači iz operativne memorije i smešta u ulaz memorije DATA koji je određen za zamenu. Takođe se, broj bloka operativne memorije, određen sa najviših *h* bita generisane adrese, upisuje u memoriju TAG i to u ulaz određen za zamenu.

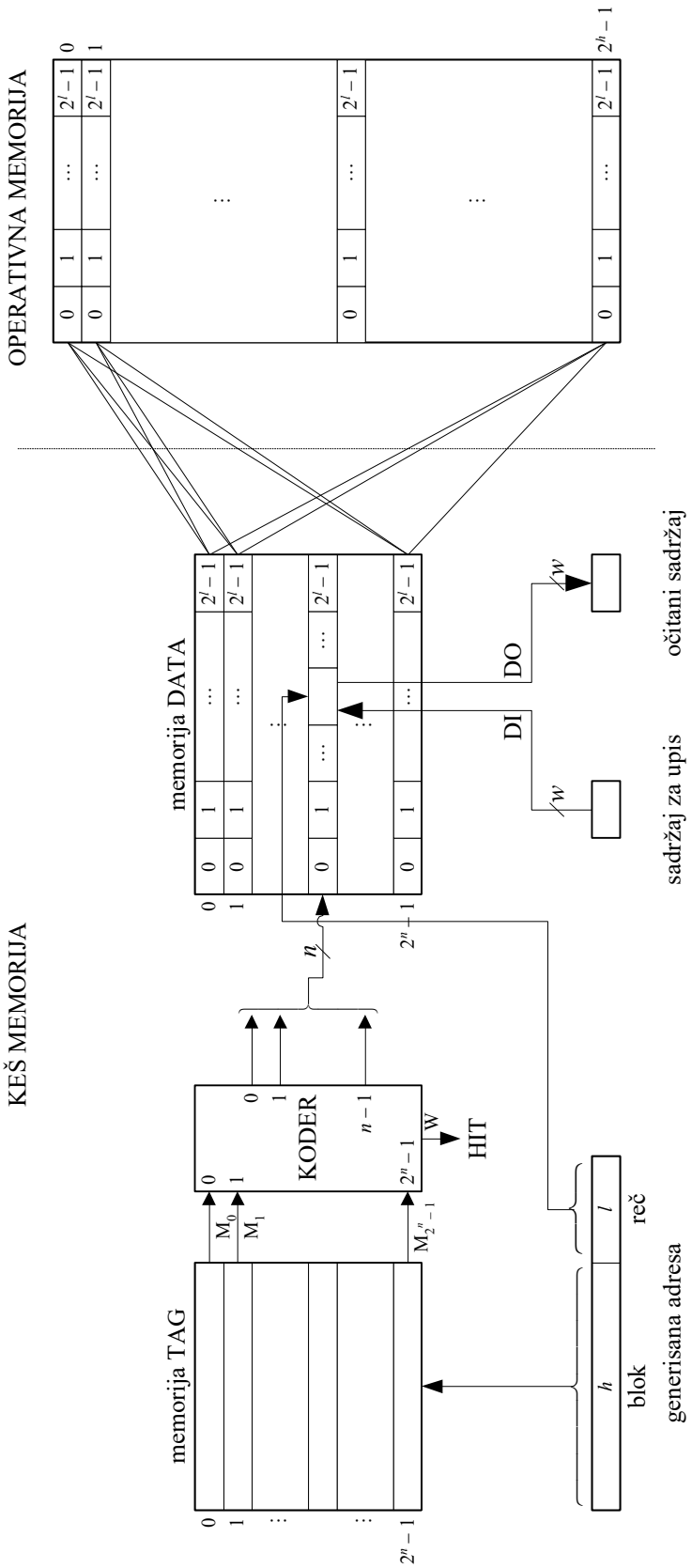
Pri generisanju zahteva za upis od strane procesora na isti način se ispituje saglasnost sa sadržajem keš memorije kao u slučaju operacije čitanja. Ukoliko postoji saglasnost, sa *n* bita sa izlaza kodera i *l* bita generisane adrese, adresira se reč memorije DATA i obavlja upis. Ako saglasnost ne postoji, na identičan način kao i za operaciju čitanja se, najpre, blok ulaza keš memorije određenog za zamenu prebacuje u operativnu memoriju, a zatim u isti ulaz keš memorije dovlači novi blok iz operativne memorije. Potom se, ponovo, na već opisani način vrši provera da li postoji saglasnost, utvrđuje da postoji saglasnost i realizuje upis.

Opisani mehanizam čitanja iz i upisa u keš memoriju je osnovni mehanizam rada sa keš memorijom. U praktičnim realizacijama postoji više varijanti ovog osnovnog mehanizma.

Prednost ove tehnike je pre svega u boljem popunjavanju keš memorije jer bilo koji blok operativne memorije može da se smesti u bilo koji ulaz keš memorije. Nedostatak je u potrebi da se implementira neki od algoritama zamene i visoka cena asocijativne memorije koja se koristi u realizaciji memorije TAG.

1.1.1.3. SET-ASOCIJATIVNO PRESLIKAVANJE

Tehnika set-asocijativnog preslikavanja predstavlja kombinaciju prethodne dve tehnike preslikavanja. Može se uzeti da je set-asocijativno preslikavanje dobijeno prostom podelom keš memorije na dve ili više identičnih manjih keš memorija. Njihov broj je 2^b, gde je *b* prirodan broj, i svaka od njih je realizovana tehnikom direktnog preslikavanja. Operativna memorija se sada može podeliti na više grupa tako da broj blokova u grupi odgovara broju blokova jedne manje keš memorije. Kao i u tehnici direktnog preslikavanja na osnovu broja bloka u grupi se utvrđuje u koji ulaz manje keš memorije se dati blok preslikava. Pošto ovde ima 2^b manjih keš memorija, taj blok može da se preslika u bilo koji od njih, jedino će broj bloka u toj manjoj keš memoriji odgovarati broju bloka u grupi operativne memorije. Time je stvoreno onoliko setova keš memorije koliko ima blokova u grupi operativne memorije, sa onoliko blokova po setu keš memorije koliko ima manjih keš memorija. Na nivou seta preslikavanje je direktno, jer je brojem bloka u grupi operativne memorije jednoznačno određen set u koji se dati blok preslikava. Unutar seta preslikavanje je asocijativno, jer se unutar seta dati blok operativne memorije može smestiti u bilo koji od blokova seta.



Slika 2 Keš memorija sa asocijativnim preslikavanjem

Kod ove tehnike preslikavanja keš memorija se sastoji iz 2^b manjih keš memorija i to keš memorije blokova 0 u kojoj se za sve setove čuvaju samo blokovi 0, keš memorije blokova 1 u kojoj se za sve setove čuvaju samo blokovi 1 i tako redom do keš memorije blokova $(2^b - 1)$ u kojoj se za sve setove čuvaju samo blokovi $(2^b - 1)$. Keš memorija se sastoji od sledećih delova (slika 3):

- memorija $DATA_i$ i
- memorija TAG_i.

pri čemu je $i = 0, 1, \ldots, 2^b - 1$.

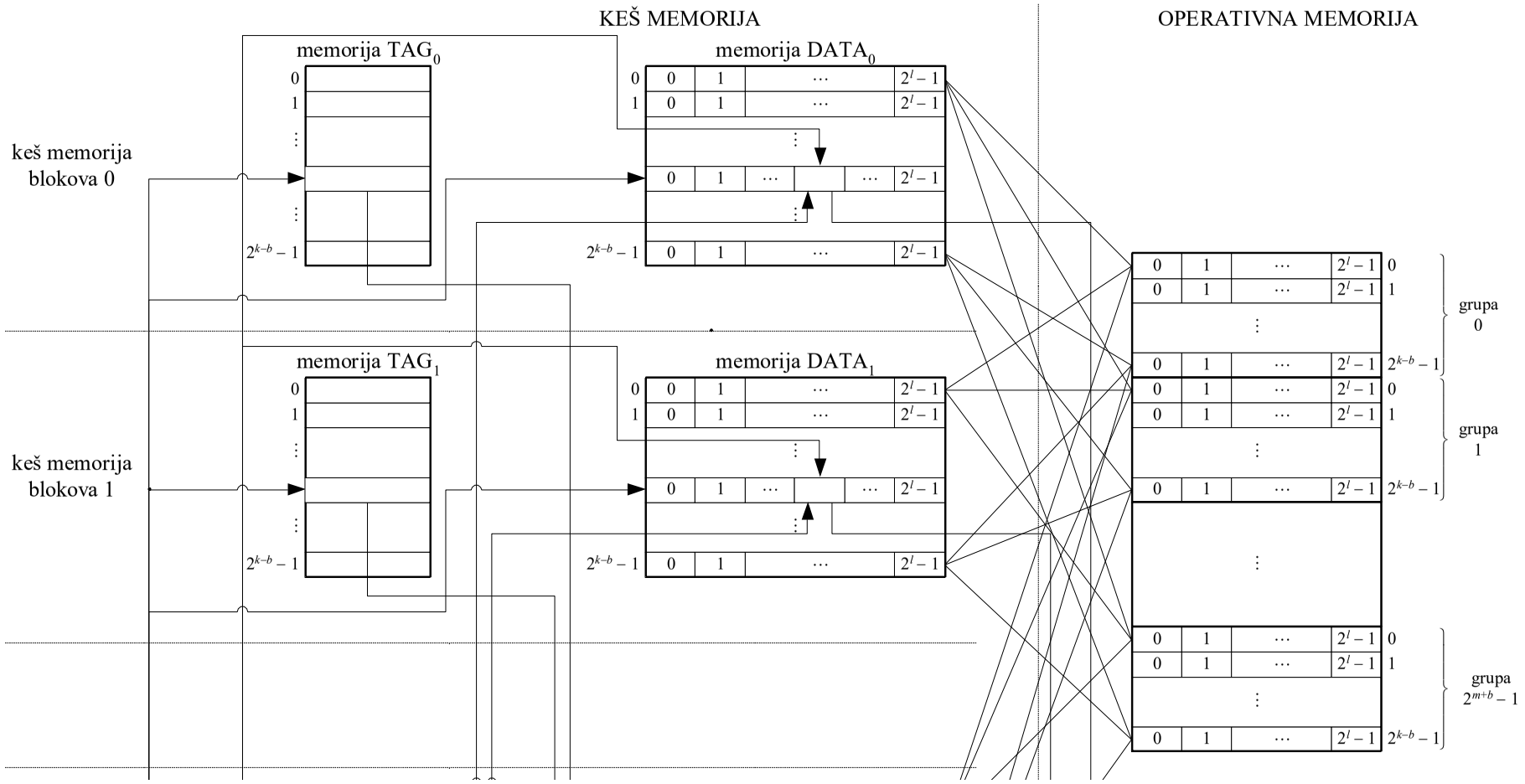
U bilo koju memoriju $DATA_i$, $i = 0, 1, \ldots, 2^b - 1$, se smeštaju blokovi preneti iz operativne u keš memoriju. U odgovarajuću memoriju TAG_i, $i = 0, 1, \ldots, 2^b - 1$, se smeštaju brojevi grupa blokova prenetih iz operativne u keš memoriju. Za realizaciju memorija $DATA_i$ i TAG_i, $i = 0, 1, \ldots, 2^b - 1$, koristi se RAM memorija. Ukoliko u memoriju $DATA_i$, $i = 0, 1, \ldots, 2^b - 1$, može da se smesti 2^{k-b} blokova operativne memorije kaže se da keš memorija ima 2^{k-b} setova. Za svaki set postoji poseban ulaz u svakoj od memorija $DATA_i$ i TAG_i, $i = 0, 1, \ldots, 2^b - 1$. Memorije $DATA_i$ i TAG_i, $i = 0, 1, \ldots, 2^b - 1$ obrazuju memorije DATA i TAG keš memorije sa set-asocijativnim preslikavanjem.

U slučaju keš memorije sa 2^{k-b} setova realizovane u tehnici set-asocijativnog preslikavanja zamišlja se kao da je operativna memorija kapaciteta 2^{m+k} blokova podeljena na 2^{m+b} grupa pri čemu veličina grupe odgovara veličini jedne manje memorije $DATA_i$, $i = 0, 1, \ldots, 2^b - 1$, i iznosi 2^{k-b} blokova. Ukoliko se uzme da je dimenzija bloka 2^l reči, tada generisana adresa ima sledeću strukturu:

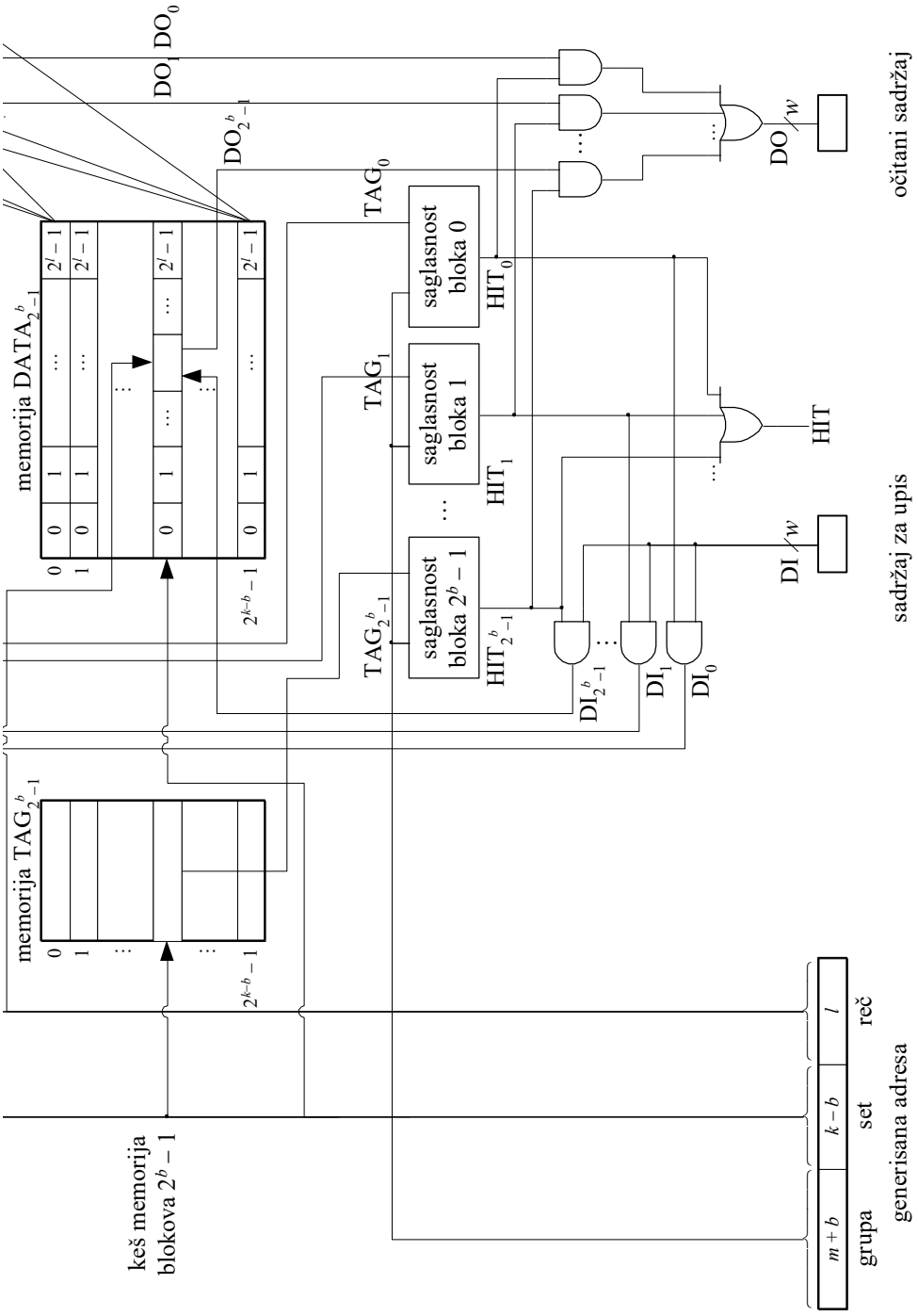
- najnižih l bita određuju adresu reči unutar bloka,
- srednjih $(k - b)$ određuju broj bloka unutar grupe i broj seta u keš memoriji i
- najviših $(m + b)$ bita određuju broj grupe u operativnoj memoriji kojoj pripada blok.

U ovoj tehnici svaki blok operativne memorije ima fiksno definisan set u keš memoriji u koji se smešta prilikom dovlačenja iz operativne u keš memoriju. Tako se j-ti blok iz bilo koje od 2^{m+b} grupa operativne memorije prilikom dovlačenja može da smesti jedino u j-ti set keš memorije. Pošto se u bilo kom od 2^b blokova j-tog seta može da nađe j-ti blok iz bilo koje od 2^{m+b} grupa operativne memorije, u j-tom ulazu jedne od 2^b memorija TAG_i, $i = 0, 1, \ldots, 2^b - 1$, se čuva broj grupe kojoj pripada dati j-ti blok. S obzirom da u operativnoj memoriji ima 2^{m+b} grupa, širina memorijske reči memorija TAG_i, $i = 0, 1, \ldots, 2^b - 1$, je $(m + b)$ bita. Totalni kapacitet keš memorije je kao i u slučaju direktnog preslikavanja 2^k blokova, ali su sada oni organizovani u 2^{k-b} setova sa 2^b blokova po setu.

Pri generisanju zahteva za čitanje od strane procesora srednjih $(k - b)$ bita generisane adrese, koji predstavljaju broj seta, određuju ulaz u svakoj od memorija TAG_i i $DATA_i$, $i = 0, 1, \ldots, 2^b - 1$. Sadržaji adresiranih lokacija memorija TAG_i, $i = 0, 1, \ldots, 2^b - 1$, porede se sa najviših $(m + b)$ bita generisane adrese da bi se utvrdilo da li u nekom od 2^b blokova datog seta postoji saglasnost ili ne. Ukoliko postoji saglasnost, sa srednjih $(k - b)$ i najnižih l bita generisane adrese adresira se reč memorije $DATA_i$, $i = 0, 1, \ldots, 2^b - 1$, onog bloka u setu za koji je otkrivena saglasnost i obavlja čitanje. Ako saglasnost ne postoji ni u jednom od 2^b blokova datog seta onda se korišćenjem nekog od algoritama zamene bira jedan od 2^b blokova seta za zamenu i prebacuje u operativnu memoriju. Broj grupe operativne memorije u koju se blok upisuje određen je sa $(m + b)$ bita očitanih iz memorije TAG_i, $i = 0, 1, \ldots, 2^b - 1$, koja odgovara bloku u setu koji je odabran za zamenu i to sa ulaza određenog sa srednjih $(k + b)$ bita generisane adrese. Broj bloka unutar date grupe operativne memorije određen je sa


KEŠ MEMORIJA
OPERATIVNA MEMORIJA
memorija TAG0
memorija DATA0
keš memorija blokova 0
memorija TAG1
memorija DATA1
keš memorija blokova 1
grupa 0
grupa 1
grupa 2m+b − 1

Slika 3a Keš memorija sa set-asocijativnim preslikavanjem



Slika 3b Keš memorija sa set-asocijativnim preslikavanjem

srednjih ($k - b$) bita generisane adrese. Potom se ceo blok u kome se nalazi adresirani sadržaj dovlači iz operativne memorije i smešta u ulaz memorije $DATA_i$, $i = 0, 1, \ldots, 2^b - 1$, koja odgovara bloku u setu koji je odabran za zamenu pri čemu je broj seta određen sa srednjih ($k - b$) bita generisane adrese. Takođe se broj grupe kojoj dati blok pripada, određen sa najviših ($m + b$) bita generisane adrese, upisuje u memoriju TAG_i, $i = 0, 1, \ldots, 2^b - 1$, koja odgovara bloku u setu koji je odabran za zamenu i to u ulaz određen sa srednjih ($k - b$) bita generisane adrese. Potom se na već opisani način vrši provera da li postoji saglasnost, utvrđuje da postoji, i realizuje čitanje.

Pri generisanju zahteva za upis od strane procesora na isti način se ispituje saglasnost sa sadržajem keš memorije kao u slučaju operacije čitanja. Ukoliko postoji saglasnost sa srednjih ($k - b$) i najnižih l bita generisane adrese, adresira se reč memorije DATA i nakon toga vrši se upis. Ako saglasnost ne postoji, na identičan način kao i za operaciju čitanja se najpre blok iz ulaza keš memorije određenog sa srednjih ($k - b$) bita generisane adrese koja odgovara bloku u setu koji je odabran za zamenu prebacuje u operativnu memoriju, a zatim u isti ulaz keš memorije dovlači novi blok iz operativne memorije. Potom se, ponovo, na već opisani način vrši provera da li postoji saglasnost, utvrđuje da postoji, i realizuje upis.

Opisani mehanizam čitanja iz i upisa u keš memoriju je osnovni mehanizam rada sa keš memorijom. U praktičnim realizacijama postoji više varijanti ovog osnovnog mehanizma.

Set-asocijativno preslikavanje otklanja nedostatke tehnika direktnog i asocijativnog preslikavanja. Ovde je u odnosu na direktno preslikavanje efikasnije popunjavanje blokova memorije DATA, a u odnosu na asocijativno preslikavanje smanjeno je asocijativno preslikavanje.

1.1.2. DIMENZIJA BLOKA

Kod projektovanja keš memorije neophodno je pažljivo odabrati dimenziju bloka jer ona može značajno da utiče na performanse keš memorije, a time i celog računara. Blokovi manjih dimenzija imaju neke prednosti. Vreme neophodno za prenošenje ovakvog bloka iz operativne memorije u keš memoriju i obrnuto je kraće od onog potrebnog za veći blok. Ako je računar projektovan tako da mora da se sačeka da se prenese ceo blok pa tek onda da se pristupa datom bloku u keš memoriji, manji blokovi su povoljniji od većih. Sledeća prednost blokova manjih dimenzija je u tome što je manja verovatnoća da ovakav blok sadrži puno nepotrebnih reči. Širina reči u operativnoj memoriji predstavlja donju granicu za dimenziju bloka.

Blokovi većih dimenzija takođe imaju određene prednosti. Što je dimenzija bloka veća, tada je za određeni kapacitet keš memorije, manji broj blokova u keš memoriji, a samim tim manja je i logika neophodna za implementaciju algoritma zamene, kao i ostala neophodna logika, koja je vezana za svaki od blokova. Manji je i deo adrese koji se upisuje u memoriju TAG, a i sama memorija TAG je manjeg kapaciteta. Ako je u pitanju program koji se sekvencijalno izvršava ili se pristupa sekvencijalnoj strukturi podataka, efikasnije je da se što veći deo programa ili date strukture podataka dovuče odjednom u keš memoriju, pa tada veća dimenzija bloka predstavlja bolje rešenje.

Potrebno je primetiti da su sve prednosti nabrojane gore za blokove manjih dimenzija u isto vreme i nedostaci za blokove većih dimenzija i obrnuto.

1.1.3. ZAMENA BLOKOVA KEŠ MEMORIJE

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

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

Pri izboru algoritma zamene treba voditi računa o dva zahteva. Prvi je da on treba da bude takav da bude minimalna verovatnoća da će blok koji je odabran za zamenu i vraćen iz keš u operativnu memoriju ubrzo morati ponovo da se dovuče iz operativne u keš memoriju. Drugi je da cena hardvera potrebnog za njegovu realizaciju bude što je moguće niža. Ova dva zahteva su kontradiktorna, pa je cena hardvera algoritama zamene koji bolje ispunjavaju prvi zahtev viša i obrnuto.

Kod korišćenih algoritama zamene ima više pristupa. Jedan je da se zameni onaj blok kome se najduže vremena nije pristupalo. Takav blok se definiše kao LRU (*least recently used*) blok, a algoritam zamene se naziva LRU algoritam. Drugi pristup je da se zameni blok koji je najranije unet u keš memoriju. Takav blok se definiše kao FIFO (*first in—first out*) blok, a algoritam zamene se naziva FIFO algoritam. Hardver za implementaciju FIFO algoritma je dosta jednostavniji od hardvera potrebnog za realizaciju LRU algoritma. Interesantno je da se dosta često primenjuje i algoritam kojim se slučajno bira blok za zamenu korišćenjem jednog od postojećih generatora slučajnih brojeva.

1.1.4. AŽURIRANJE OPERATIVNE MEMORIJE

Pri svakom zahtevu za upis modifikuje se sadržaj u nekom od blokova u keš memoriji. Zbog tih promena koje nastaju u kopijama blokova operativne memorije koje se nalaze u keš memoriji, potrebno je u nekom trenutku izvršiti njihovo ažuriranje i u operativnoj memoriji. Postoje dva pristupa kojima se obezbeđuje da sadržaji u operativnoj memoriji budu ažurirani, i to:

1. *upiši skroz*, koji se u engleskoj terminologiji naziva *write-through* ili *store-through* i
2. *vrati nazad*, koji se u engleskoj terminologiji naziva *write-back* ili *copy-back.*

Kod pristupa *upiši skroz* pri svakom zahtevu za upis upisivanje se istovremeno vrši i u keš memoriju i u operativnu memoriju. Time se obezbeđuje da sadržaj u operativnoj memoriji bude uvek ažuran.

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

operativnu memoriju. Stoga kod keš memorija koje koriste ovaj pristup ažuriranja sadržaja operativne memorije, pored zahteva za čitanje i upis, postoji i zahtev za čišćenje keš memorije (*flush*).

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

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

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

1.1.5. NEKA RAZMATRANJA U VEZI REALIZACIJE KEŠ MEMORIJE

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

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

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

Sledeće poboljšanje je moguće ostvariti u slučajevima kada je potrebno izvršiti vraćanje modifikovanog bloka u operativnu memoriju. Da bi se ubrzao taj postupak moguće je postaviti bafer koji će prihvatiti ceo blok što bi omogućilo da se odmah pređe na dovlačenje novog bloka iz operativne memorije. Tek po završetku dovlačenja novog bloka prelazi se na vraćanje u operativnu memoriju bloka koji se nalazi u baferu. I ovde keš memorija ne može prihvatiti novi zahtev za čitanje ili upis sve dok se cela operacija ne završi do kraja. Ovo poboljšanje se naziva *baferisanje*.

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

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

1.2. REALIZACIJA KEŠ MEMORIJA

U ovom poglavlju se daju realizacije keš memorija sa direktnim, asocijativnim i set-asocijativnim preslikavanjem. U svim realizacijama veličina bloka je 4 bajta. Kod direktnog i set-asocijativnog preslikavanja se koristi *vrati nazad*, a kod asocijativnog preslikavanja *upiši skroz* tehnika ažuriranja sadržaja operativne memorije. Kod set-asocijativnog preslikavanja blok podataka koji se vraća iz keš memorije u operativnu memoriju se baferuje, a kod asocijativnog preslikavanja bajt podatka koji se upisuje se baferuje. Za zamenu bloka keš memorije se koristi LRU algoritam kod asocijativnog preslikavanja, a FIFO algoritam kod set-asocijativnog preslikavanja.

1.2.1. KEŠ MEMORIJA SA DIREKTNIM PRESLIKAVANJEM

Keš memorija koja se razmatra realizovana je tehnikom direktnog preslikavanja na nivou bloka veličine 4 bajta. U ovoj tehnici preslikavanja svaki blok operativne memorije ima fiksno definisano mesto u keš memoriji, pa ovde nije potrebno definisati algoritam zamene. Za ažuriranja operativne memorije uzeta je metoda *vrati-nazad*.

Keš memorija ima 8 ulaza u kojima se čuva 8 blokova operativne memorije. S obzirom da je blok veličine 4 bajta, kapacitet dela keš memorije u kome se čuva sadržaj je 32 bajta, a adresibilna jedinica je jedan bajt.

Operativna memorija je kapaciteta 64 Kbajta, a adresibilna jedinica je jedan bajt. Stoga se operativna memorija može posmatrati kao da je organizovana u 2^{11} grupa, od kojih svaka sadrži 2^3 blokova veličine 2^2 bajta. Adresa operativne memorije dužine 16 bita može se podeliti i označiti na sledeći način:

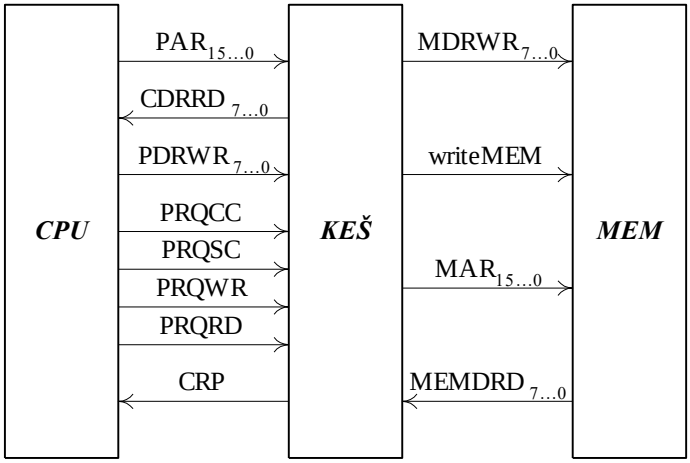
- najviših 11 bitova označavaju broj grupe,
- srednja 3 bita označavaju broj bloka i
- najniža 2 bita označavaju adresu bajta u bloku.

Keš memorija je deo sistema (slika 4) koji se sastoji iz:

- procesora ***CPU***,
- memorije ***MEM*** i
- keš memorije ***KEŠ***.

Uzeto je da su sve sekvencijalne mreže sistema sinhrone i da ceo sistem radi sinhrono sa zajedničkim signalom takta **CLK**.

U ovom poglavlju daju se detalji realizacije procesora ***CPU*** i memorije ***MEM*** relevantni za rad keš memorije ***KEŠ*** i kompletna hardverska realizacija keš memorije ***KEŠ***.



Slika 4 Struktura sistema

1.2.1.1. PROCESOR CPU

Procesor ***CPU*** se obraća keš memoriji ***KEŠ*** onda kada treba očitati podatak iz keš memorije ***KEŠ***, upisati podatak u keš memoriju ***KEŠ***, vratiti iz keš memorije ***KEŠ*** u memoriju ***MEM*** određeni blok podataka ukoliko je blok modifikovan ili vratiti iz keš memorije ***KEŠ*** u memoriju ***MEM*** sve blokove podataka koji su modifikovani. Ova četiri obraćanja procesora ***CPU*** keš memoriji ***KEŠ*** se u daljem tekstu nazivaju operacija čitanja, operacija upisa, operacija selektivnog vraćanja i operacija kompletnog vraćanja.

Operacije čitanja ili upisa se koriste kad god se generiše adresa memorije ***MEM*** sa koje treba očitati ili instrukciju ili operand ili na kojoj treba upisati rezultat.

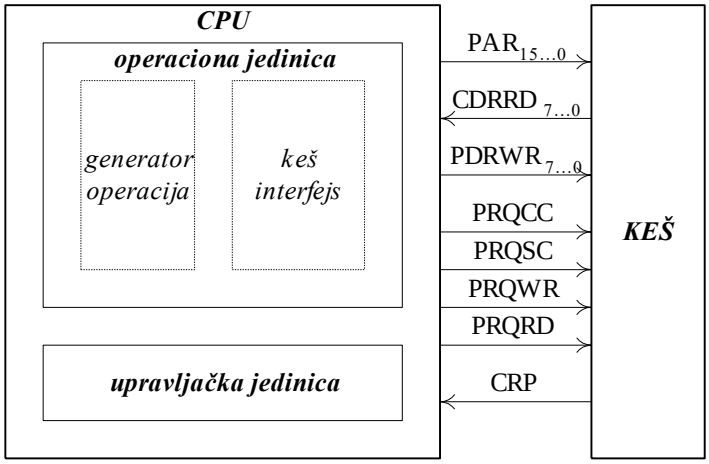
Operacija selektivnog vraćanja se koristi onda kada sve modifikovane blokove iz određenog opsega adresa memorije ***MEM*** treba vratiti iz keš memorije ***KEŠ*** u memoriju ***MEM***. Ukoliko taj opseg adresa sadrži *n* blokova, procesor ***CPU*** generiše *n* operacija selektivnog vraćanja. Svaka operacija selektivnog vraćanja specificira i adresu jednog od bajtova bloka. Kod ove operacije keš memorija ***KEŠ*** vrši proveru da li se podatak sa zadate adrese nalazi u odgovarajućem bloku keš memorije ***KEŠ***. U slučaju da se nalazi, vrši se provera da li je blok modifikovan. Ukoliko jeste, blok se vraća iz keš memorije ***KEŠ*** u memoriju ***MEM*** i operacija završava. Ukoliko nije, operacija se odmah završava. U slučaju da se blok ne nalazi u keš memoriji ***KEŠ***, operacija se završava.

Operacija kompletnog vraćanja se koristi onda kada treba sve modifikovane blokove keš memorije ***KEŠ*** vratiti iz keš memorije ***KEŠ*** u memoriju ***MEM***. Kod ove operacije nema specificiranja adrese, jer se vrši provera redom za sve blokove keš memorije ***KEŠ*** da li su modifikovani. Onaj blok za koji se utvrdi da je modifikovan vraća se iz keš memorije ***KEŠ*** u memoriju ***MEM***, pa se prelazi na proveru sledećeg bloka. Onaj blok za koji se utvrdi da nije modifikovan ne dira se, već se odmah prelazi na proveru sledećeg bloka. Operacija kompletnog vraćanja se završava pošto se opisani postupak realizuje za sve blokove keš memorije ***KEŠ***.

Operacije selektivnog i kompletnog vraćanja se koriste kod softverskog održavanja koherencije operativne memorije i keš memorije i to u slučaju kada se za ažuriranje sadržaja operativne memorije koristi tehnika *vrati nazad*. Operacije selektivnog vraćanja se koriste

kada to treba uraditi samo za određeni opseg adresa operativne memorije, a operacija kompletnog vraćanja se koristi kada to treba uraditi za kompletan opseg adresa operativne memorije.

Signali koji se razmenjuju između procesora ***CPU*** i keš memorije ***KEŠ*** prilikom izvršavanja ove četiri operacije prikazani su na slici 5.



Slika 5 Procesor CPU

Kod operacije čitanja procesor ***CPU*** šalje keš memoriji ***KEŠ*** 16-bitnu adresu po linijama **PAR$_{15...0}$** i generiše aktivnu vrednost signala **PRQRD** trajanja jedna perioda signala takta. Očitani 8-bitni podatak se vraća po linijama **CDRRD$_{7...0}$**. Operacija čitanja je završena i na linijama **CDRRD$_{7...0}$** je važeći podatak onda kada keš memorija ***KEŠ*** generiše aktivnu vrednost signala **CRP** trajanja jedna perioda signala takta.

Kod operacije upisa procesor ***CPU*** šalje keš memoriji ***KEŠ*** 16-bitnu adresu po linijama **PAR$_{15...0}$**, 8-bitni podatak za upis po linijama **PDRWR$_{7...0}$** i generiše aktivnu vrednost signala **PRQWR** trajanja jedna perioda signala takta. Operacija upisa je završena onda kada keš memorija ***KEŠ*** generiše aktivnu vrednost signala **CRP** trajanja jedna perioda signala takta.

Kod operacije selektivnog vraćanja procesor ***CPU*** šalje keš memoriji ***KEŠ*** 16-bitnu adresu po linijama **PAR$_{15...0}$** i generiše aktivnu vrednost signala **PRQSC** trajanja jedna perioda signala takta. Operacija selektivnog vraćanja je završena onda kada keš memorija ***KEŠ*** generiše aktivnu vrednost signala **CRP** trajanja jedna perioda signala takta.

Kod operacije kompletnog vraćanja procesor ***CPU*** generiše aktivnu vrednost signala **PRQCC** trajanja jedna perioda signala takta. Operacija kompletnog vraćanja je završena onda kada keš memorija ***KEŠ*** generiše aktivnu vrednost signala **CRP** trajanja jedna perioda signala takta.

Procesor ***CPU*** (slika 5) 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 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 operacija procesora ***CPU*** i signala logičkih uslova.

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

1.2.1.1.1. Operaciona jedinica

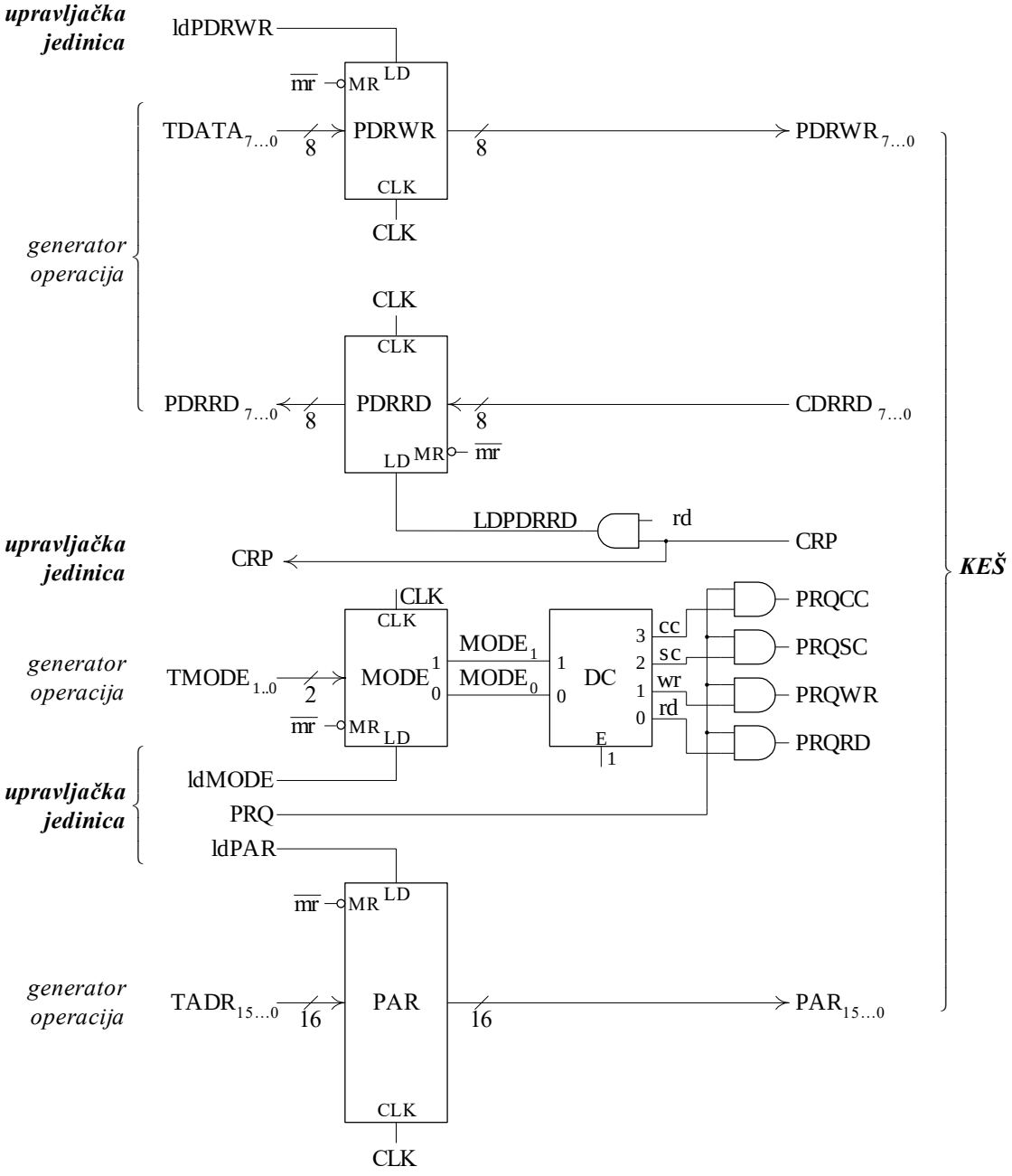
Operaciona jedinica (slika 5) se sastoji iz sledećih blokova:

- blok *keš interfejs* i
- blok *generator operacija*.

Blok *keš interfejs* služi za povezivanje procesora ***CPU*** i keš memorije ***KEŠ***. Blok *generator operacija* služi za generisanje operacija čitanja, upisa, selektivnog vraćanja i kompletnog vraćanja. Struktura i opis blokova ***operacione jedinice*** se daju u daljem tekstu.

1.2.1.1.1.1. Blok keš interfejs

Blok *keš interfejs* (slika 6) sadrži registre PAR, PDRWR, PDRRD i MODE, dekoder DC i 4 logička I kola.



Slika 6 Blok keš interfejs

Registar PAR (Processor Address Register) služi za čuvanje ili adrese lokacije memorije ***MEM*** sa koje treba očitati sadržaj u slučaju operacije čitanja ili adrese lokacije memorije ***MEM*** u koju treba upisati sadržaj u slučaju operacije upisa ili adrese lokacije memorije ***MEM*** koja pripada bloku koji treba, ukoliko je modifikovan, vratiti iz keš memorije ***KEŠ*** u memoriju ***MEM*** u slučaju operacije selektivnog vraćanja. Izlazne linije ovog registra se vode kao 16 adresnih linija $\mathbf{PAR_{15...0}}$ u keš memoriju ***KEŠ***. Pretpostavlja se da je procesor ***CPU*** pre obraćanja keš memoriji ***KEŠ*** generisanjem aktivne vrednosti signala **ldPAR** trajanja jedna perioda signala takta na signal takta izvršio upis adrese sa linija $\mathbf{TADR_{15...0}}$ u registar PAR.

Registar PDRWR (Processor Data Register for WRite) služi za čuvanje sadržaja koji treba upisati kod operacije upisa. Izlazne linije ovog registra se vode kao 8 linija podataka $\mathbf{PDRWR_{7...0}}$ u keš memoriju ***KEŠ***. Pretpostavlja se da je procesor ***CPU*** pre obraćanja keš memoriji ***KEŠ*** već upisao sadržaj u registar PDRWR, tako što je generisao aktivnu vrednost

signala **ldPDRWR** trajanja jedna perioda signala takta i na signal takta izvršio upis sadržaja sa linija $\mathbf{TDATA_{7...0}}$.

Registar PDRRD (Processor Data Register for ReaD) služi za čuvanje sadržaja koji je očitan kod operacije čitanja. Očitani podatak se iz keš memorije ***KEŠ*** vodi po linijama $\mathbf{CDRRD_{7...0}}$ na ulaze registra PDRRD. U slučaju operacije čitanja signal **rd** je aktivan, pa se aktivnom vrednošću signala **CRP** trajanja jedna perioda signala takta na signal takta izvrši upis očitanog sadržaja sa linija $\mathbf{CDRRD_{7...0}}$ u registar PDRRD.

Registar MODE (operation MODE) služi za čuvanje binarne vrednosti jedne od četiri operacije koje keš memorija ***KEŠ*** može da realizuje. Pretpostavlja se da je procesor ***CPU*** pre obraćanja keš memoriji ***KEŠ*** već upisao binarnu vrednost operacije u registar MODE, tako što je generisao aktivnu vrednost signala **ldMODE** trajanja jedna perioda signala takta i najkasnije na signal takta kojim se signal **PRQ** (Processor ReQuest) postavlja na aktivnu vrednost izvršio upis sadržaja sa linija $\mathbf{TMODE_{1...0}}$ u registar MODE. Signali $\mathbf{MODE_{1...0}}$ sa izlaza registra MODE se vode na ulaze dekodera DC.

Dekoder DC i četiri logička I kola služe za generisanje signala **PRQRD**, **PRQWR**, **PRQSC** i **PRQCC**. Dekoder DC na svojim izlazima 0, 1, 2 i 3 daje aktivnu vrednost jednog od signala **rd**, **wr**, **sc** i **cc** u zavisnosti od binarne vrednosti signala $\mathbf{MODE_{1...0}}$ sa ulaza. Pri aktivnoj vrednosti signala **PRQ** trajanja jedna perioda signala takta jedan od signala **PRQRD**, **PRQWR**, **PRQSC** i **PRQCC** postaje aktivan u zavisnosti od toga koji je od signala **rd**, **wr**, **sc** i **cc** aktivan. Signali **PRQRD**, **PRQWR**, **PRQSC** i **PRQCC** su neaktivni pri neaktivnoj vrednosti signala **PRQ**.

Upravljački signal **PRQRD** (Processor ReQuest for ReaD) generiše procesor ***CPU*** da, aktivnom vrednošću ovog signala trajanja jedna perioda signala takta, signalizira keš memoriji ***KEŠ*** da se na linijama $\mathbf{PAR_{15...0}}$ nalazi adresa lokacije sa koje treba očitati podatak i da u keš memoriji ***KEŠ*** startuje operaciju čitanja.

Upravljački signal **PRQWR** (Processor ReQuest for WRite) generiše procesor ***CPU*** da, aktivnom vrednošću ovog signala trajanja jedna perioda signala takta, signalizira keš memoriji ***KEŠ*** da se na linijama $\mathbf{PAR_{15...0}}$ nalazi adresa lokacije u koju treba upisati podatak sa linija $\mathbf{PDRWR_{15...0}}$ i da u keš memoriji ***KEŠ*** startuje operaciju upisa.

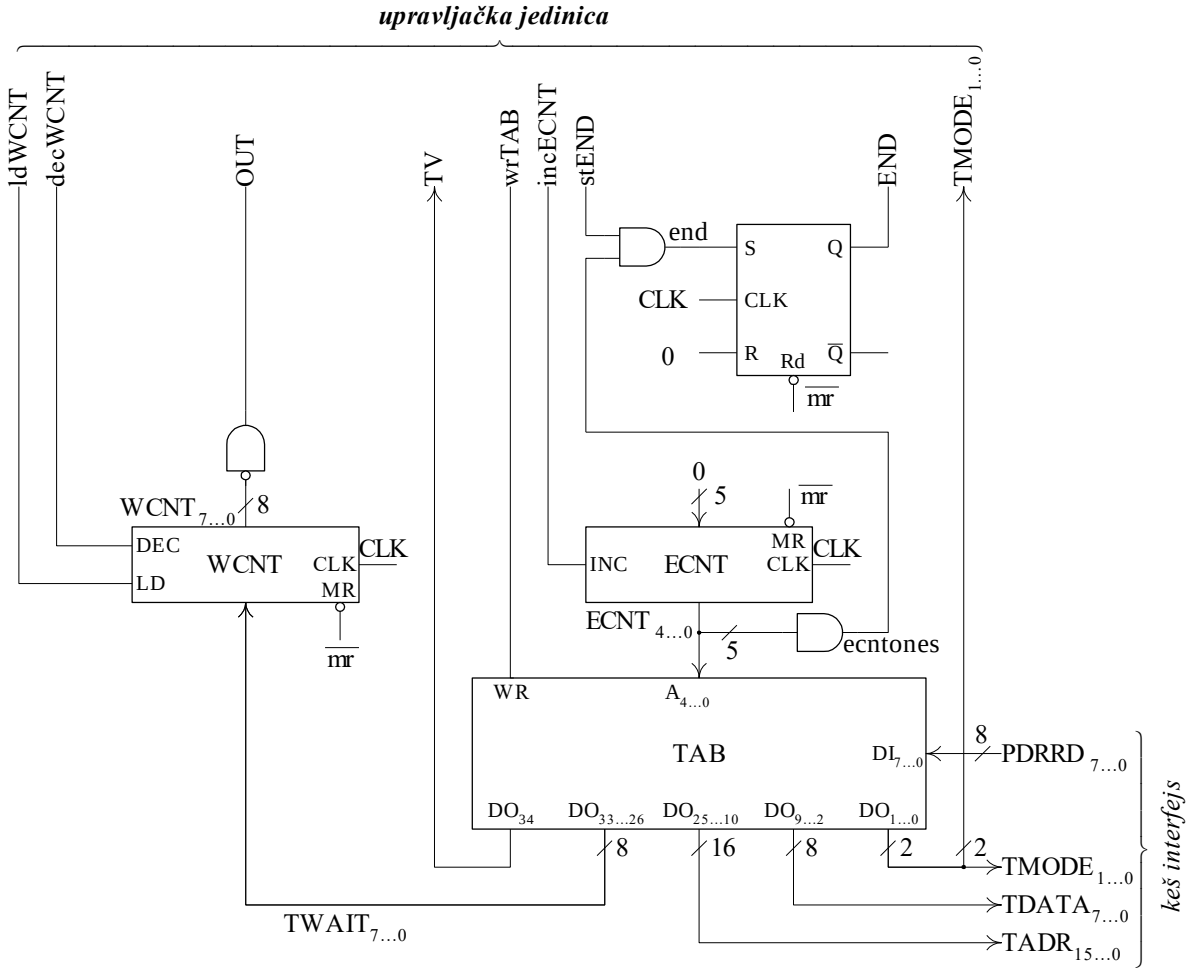
Upravljački signal **PRQSC** (Processor ReQuest for Selective Clear) generiše procesor ***CPU*** da, aktivnom vrednošću ovog signala trajanja jedna perioda signala takta, signalizira keš memoriji ***KEŠ*** da se na linijama $\mathbf{PAR_{15...0}}$ nalazi adresa lokacije za koju treba realizovati operaciju selektivnog vraćanja i da u keš memoriji ***KEŠ*** startuje operaciju selektivnog vraćanja.

Upravljački signal **PRQCC** (Processor ReQuest for Complete Clear) generiše procesor ***CPU*** da, aktivnom vrednošću ovog signala trajanja jedna perioda signala takta, u keš memoriji ***KEŠ*** startuje operaciju kompletnog vraćanja.

Upravljački signal **CRP** (Cache RePly) generiše keš memorija ***KEŠ*** da, aktivnom vrednošću ovog signala trajanja jedna perioda signala takta, signalizira procesoru ***CPU*** da je operacija čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja završena, da procesor ***CPU*** može da produži sa radom i da može u keš memoriji ***KEŠ*** da startuje neku od četiri operacije. U slučaju operacije čitanja aktivnom vrednošću signala **CRP** se signalizira i da se na linijama $\mathbf{CDRRD_{7...0}}$ nalazi očitani podatak i da aktivnom vrednošću signala **CRP** treba na signal takta sadržaj sa linija $\mathbf{CDRRD_{7...0}}$ upisati u registar PDRRD.

1.2.1.1.1.2. Blok generator operacija

Blok *generator operacija* (slika 7) sadrži brojače ECNT i WCNT, flip-flop END i memoriju TAB.



Slika 7 Blok generator operacija

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

Brojač WCNT (Wait CouNTer) služi za realizaciju čekanja između generisanja dve operacije. Dužina čekanja zavisi od sadržaja polja WAIT ulaza u memoriji TAB adresiranog sadržajem brojača WCNT (slika 8). Vrednost dužine čekanja se iz polja WAIT odgovarajućeg ulaza memorije TAB kao signali **$TWAIT_{7...0}$** vodi na paralelne ulaze brojača WCNT i upisuje u njega na signal takta pri aktivnoj vrednosti signala **ldWCNT**. Dekrementiranje brojača WCNT se realizuje na signal takta pri aktivnoj vrednosti signala **decWCNT**. Kada sadržaj brojača WCNT postane nula signal **OUT** dobija aktivnu vrednost.

Flip-flop END (generation of operations ENDed) služi kao indikacija da li se završilo sa generisanjem operacija iz svih ulaza memorije TAB. Ovaj flip-flop je na početku neaktivan, a postavlja se na signal takta 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 postane sve jedinice, što znači da se prešlo na generisanje operacije iz zadnjeg ulaza memorije TAB. Signal **stEND** postaje aktivan, u trajanju jedna perioda signala takta, u zadnjem koraku generisanja operacije iz svakog ulaza memorije TAB. Stoga signal **end** postaje aktivan u trajanju jedna perioda signala takta tek u zadnjem koraku generisanja operacije iz zadnjeg ulaza tabele TAB, pa se tada na signal takta upisuje aktivna vrednost u flip-flop END.

Memorija TAB ima 32 ulaza, pa najviše 32 operacije keš memorije ***KEŠ*** mogu da budu generisane. Jedan ulaz sadrži informacije na osnovu kojih se generiše jedna operacija keš memorije ***KEŠ*** (slika 8).

Slika 8 Struktura jednog ulaza u memoriji TAB

Bit V (Valid) označava da li je odgovarajući ulaz memorije TAB važeći. Procesor ***CPU*** kreće sa generisanjem operacija od ulaza 0 memorije TAB i zaustavlja se na onom ulazu na kome otkrije da bit V ima vrednost nula. Bit V se pojavljuje na liniji DO_{34} i označen je kao signal **TV**.

Polje WAIT (WAIT between operations) 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 nove operacije na osnovu sadržaja očitanog ulaza. Bitovi WAIT se pojavljuju na linijama $DO_{33...26}$ i označeni su kao signali $\mathbf{TWAIT_{7...0}}$.

Polje ADR (ADdRess) označava adresu memorije ***MEM*** sa koje treba očitati podatak u slučaju operacije čitanja, adresu memorije ***MEM*** na kojoj treba upisati podatak u slučaju operacije upisa i adresu memorije ***MEM*** koja pripada bloku koji treba vratiti iz keš memorije ***KEŠ*** u memorije ***MEM*** ako je blok modifikovan u slučaju operacije selektivnog vraćanja. Bitovi ADR se pojavljuju na linijama $DO_{25...10}$ i označeni su kao signali $\mathbf{TADR_{15...0}}$.

Polje DATA (DATA to be written or read) predstavlja podatak koji se upisuje u memoriju ***MEM*** u slučaju operacije upisa, odnosno očitani podatak iz memorije ***MEM*** u slučaju operacije čitanja, dok se kod preostale dve operacije ne koristi. Bitovi DATA se u slučaju operacije upisa pojavljuju na linijama $DO_{9...2}$ i označeni su kao signali $\mathbf{TDATA_{7...0}}$. Bitovi DATA se u slučaju operacije čitanja postavljaju na osnovu sadržaja sa linija $DI_{7...0}$ po kojima dolaze signali $\mathbf{PDRRD_{7...0}}$.

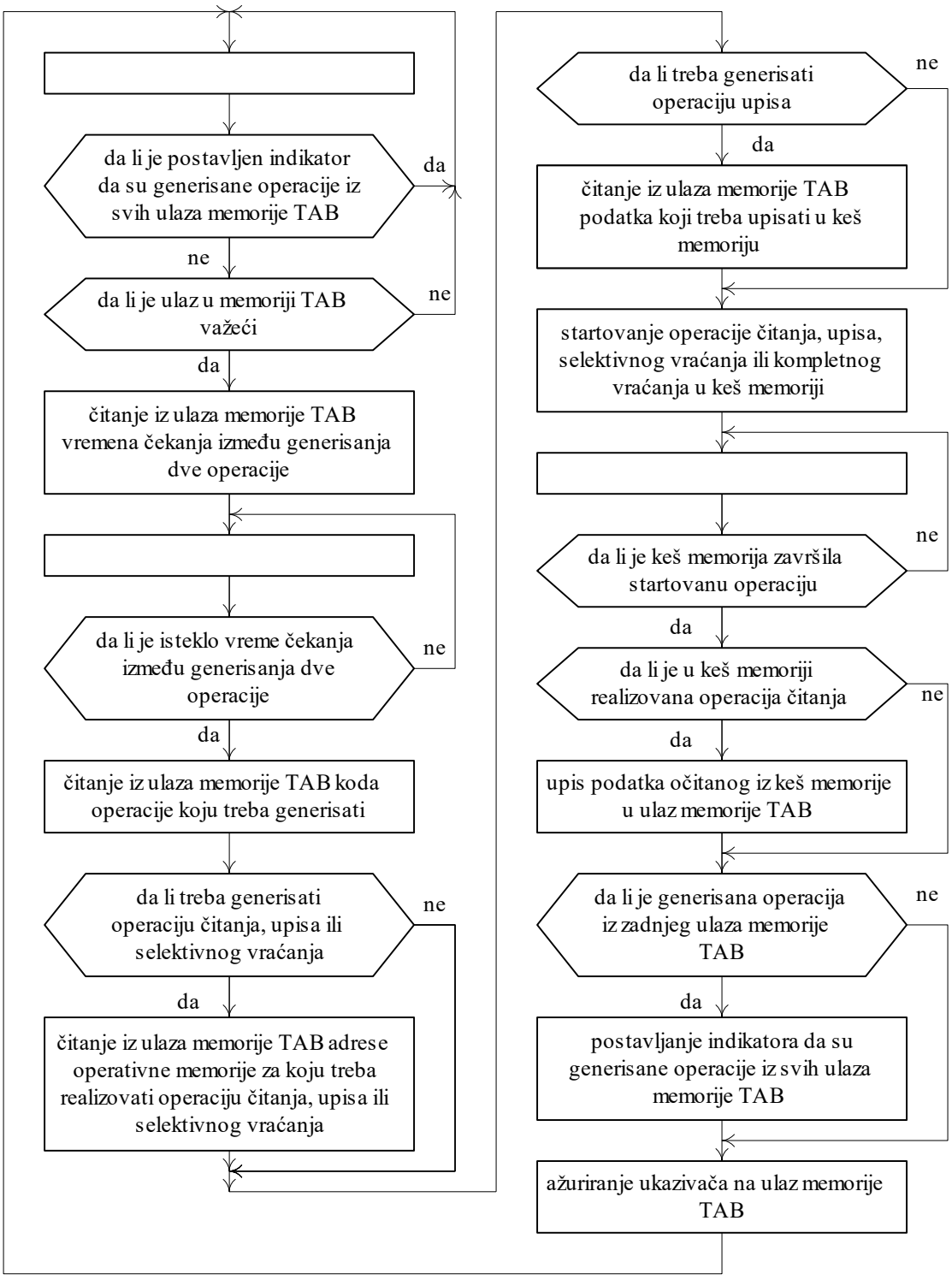
Polje MODE (MODE of operation) predstavlja kod za operaciju koju treba da realizuje keš memorija ***KEŠ***. Vrednostima od 0 do 3 označene su operacija čitanja, operacija upisa, operacija selektivnog vraćanja i operacija kompletnog vraćanja, respektivno. Bitovi MODE se pojavljuju na linijama $DO_{1...0}$ i označeni su kao signali $\mathbf{TMODE_{1...0}}$.

1.2.1.1.2. Upravljačka jedinica

U ovom poglavlju se prikazuju dijagram toka generisanja operacija, algoritam generisanja upravljačkih signala, vremenski oblici signala i struktura ***upravljačke jedinice***.

1.2.1.1.2.1. Dijagram toka generisanja operacija

Dijagram toka generisanja operacija čitanja, upisa, selektivnog vraćanja i kompletnog vraćanja je dat na slici 9.



Slika 9 Dijagram toka generisanja operacija

U početnom koraku se vrši provera da li je postavljen indikator da su generisane operacije iz svih ulaza memorije TAB. Ukoliko je indikator postavljen ostaje se u početnom koraku i nema generisanja operacija. 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 operacija. Ukoliko je ulaz važeći prelazi se na korake generisanja jedne od četiri operacije. Iz ulaza memorije TAB se, najpre, čita vreme čekanja između generisanja dve operacije, a po isteku ovog vremena i kod

operacije koju treba generisati. U slučaju operacija čitanja, upisa i selektivnog vraćanja iz ulaza memorije TAB se čita i adresa operativne memorije za koju treba datu operaciju realizovati, a u slučaju operacije upisa čita se i podatak koji treba upisati u keš memoriju. Potom se odgovarajuća operacija startuje u keš memoriji. Po prijemu indikacije da je keš memorija završila datu operaciju procesor produžava sa preostalim koracima. Najpre se u slučaju da je u keš memoriji realizovana operacija čitanja podatak očitan iz keš memorije upisuje u ulaz memorije TAB. Potom se u slučaju da je generisana operacija bila operacija generisana iz zadnjeg ulaza memorije TAB postavlja indikator da su generisane operacije iz svih ulaza memorije TAB. Na kraju se vrši ažuriranje ukazivača na ulaz memorije TAB i prelazi na početni korak.

1.2.1.1.2.2. Algoritam generisanja upravljačkih signala

Algoritam generisanja upravljačkih signala je formiran na osnovu strukture ***operacione jedinice*** (poglavlje 1.2.1.1.1) i dijagrama toka generisanja operacija čitanja, upisa, selektivnog vraćanja i kompletnog vraćanja (poglavlje 1.2.1.1.2.1). 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 operaciona jedinica.

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

<prazno> :=

Algoritam generisanja upravljačkih signala je dat u daljem tekstu.

$step_0$: *if* (**TV** · $\overline{\textbf{END}}$, **ldWCNT**)

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

! Od koraka $step_0$ se kreće prilikom generisanja svake nove operacije. U korak $step_0$ se dolazi iz koraka $step_6$ po kompletiranju generisanja zadnje operacije. U koraku $step_0$ se proverava da li signali **TV** i $\overline{\textbf{END}}$ bloka *generisanje operacija* imaju aktivne vrednosti što se dešava u slučaju kada je odgovarajući ulaz memorije TAB bloka *generisanje operacija* važeći i kada još uvek nije generisana operacija za zadnji ulaz memorije TAB, respektivno. U slučaju da signali **TV** i $\overline{\textbf{END}}$ imaju aktivne vrednosti generiše se aktivna vrednost upravljačkog signala **ldWCNT** bloka *generisanje operacija*. Signalom **ldWCNT** trajanja jedna perioda signala takta se na signal takta u brojač WCNT upisuje vrednost iz polja WAIT adresiranog ulaza memorije TAB i prelazi na korak $step_1$. U suprotnom slučaju ostaje se u koraku $step_0$.

step$_1$: *if* ($\overline{\textbf{OUT}}$, **decWCNT**),
if (**OUT**, **ldMODE**),
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 *generisanje operacija*. Neaktivna vrednost signala **OUT** označava da brojač WCNT još uvek nije došao do nule. U tom slučaju se generiše aktivna vrednost signala **decWCNT** bloka *generisanje operacija* 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 **ldMODE** bloka *keš interfejs* trajanja jedna perioda signala takta i njome u registar MODE upisuje vrednost iz polja MODE ulaza memorije TAB bloka *generisanje operacija*. Pri aktivnoj vrednosti signala **OUT** prelazi se na korak step$_2$.

step$_2$: *if* (($\overline{\textbf{TMODE}_1}$ · $\overline{\textbf{TMODE}_0}$ + $\overline{\textbf{TMODE}_1}$ · **TMODE$_0$** +
T**MODE$_1$** · $\overline{\textbf{TMODE}_0}$), **ldPAR**),
if (($\overline{\textbf{TMODE}_1}$ · **TMODE$_0$**), **ldPDRWR**),
br step$_3$

! U korak step$_2$ se dolazi iz koraka step$_1$. U njemu se vrši provera da li neki od signala ($\overline{\textbf{TMODE}_1}$ · $\overline{\textbf{TMODE}_0}$), ($\overline{\textbf{TMODE}_1}$ · **TMODE$_0$**) i (**TMODE$_1$** · $\overline{\textbf{TMODE}_0}$) kojima se specificiraju operacije čitanja, upisa i selektivnog vraćanja, respektivno, ima aktivnu vrednost. Ovi signali se formiraju na osnovu signala **TMODE$_1$** i **TMODE$_0$** bloka *generisanje operacija*. Ukoliko jeste, generiše se aktivna vrednost signala **ldPAR** trajanja jedna perioda signala takta kojom se u registar PAR bloka *keš interfejs* upisuje vrednost iz polja ADR ulaza memorije TAB bloka *generisanje operacija*. Ukoliko je aktivna vrednost signala ($\overline{\textbf{TMODE}_1}$ · **TMODE$_0$**) radi se o operaciji upisa, pa se generiše aktivna vrednost signala **ldPDRWR** trajanja jedna perioda signala takta kojom se u registar PDRWR bloka *keš interfejs* upisuje vrednost iz polja DATA ulaza memorije TAB. U koraku step$_2$ se ostaje samo jadna perioda signala takta i bezuslovno se prelazi u korak step$_3$.

step$_3$: **PRQ**,
br step$_4$

! U korak step$_3$ se dolazi iz koraka step$_2$. U ovom koraku se bezuslovno generiše aktivna vrednost signala **PRQ** bloka *keš interfejs* trajanja jedna perioda signala takta. Time se u zavisnosti od sadržaja registra MODE u bloku *keš interfejs* generiše aktivna vrednost jednog od signala **PRQRD**, **PRQWR**, **PRQSC** i **PRQCC** i time startuje keš memorija ***KEŠ*** da izvrši jednu od operacija čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja, respektivno. U koraku step$_3$ se ostaje samo jedna perioda signala takta i bezuslovno se prelazi u korak step$_4$.

step$_4$: *br* (*if* **CRP** *then* step$_5$ *else* step$_4$)

! U korak step$_4$ se dolazi iz koraka step$_3$. U koraku step$_4$ se vrši provera signala **CRP** bloka *keš interfejs* koji keš memorija ***KEŠ*** šalje procesoru ***CPU***. Neaktivna vrednost signala **CRP** je indikacija da keš memorija ***KEŠ*** nije još uvek završila startovanu operaciju, dok je aktivna vrednost indikacija da je operacija završena. U slučaju da je završena operacija čitanja, očitani podatak treba upisati sa linija CDRRD u registar PDRRD bloka *keš interfejs*. Kako je za operaciju čitanja signal **rd** aktivan, pojavljivanjem aktivne vrednosti signala **CRP** trajanja jedna perioda signala takta i signal **LDPDRRD** postaje aktivan, čime se omogućava upisivanje očitanog podatka u registar PDRRD. U koraku step$_4$ se ostaje sve dok je neaktivna vrednost signal **CRP**. Kada signal **CRP** postane aktivan prelazi se na korak step$_5$.

step$_5$: *if* ($\overline{\textbf{TMODE}_1}$ · $\overline{\textbf{TMODE}_0}$, **wrTAB**),
br step$_6$

! U korak step$_5$ se dolazi iz koraka step$_4$. U ovom koraku se jedino u slučaju operacije čitanja, kada su signali $\overline{\textbf{TMODE}_1}$ i $\overline{\textbf{TMODE}_0}$ bloka *generisanje operacija* aktivni, generiše aktivna vrednost signala **wrTAB** bloka *generisanje operacija* trajanja jedna perioda signala takta. Signalom **wrTAB** se očitani podatak upisuje iz registra PDRRD bloka *keš interfejs* u polje DATA ulaza memorije TAB bloka *generisanje operacija*. U slučaju preostale tri operacije u ovom koraku se ne generiše ni jedan od upravljačkih signala. U koraku step$_5$ se ostaje samo jedna perioda signala takta i bezuslovno se prelazi u korak step$_6$.

step$_6$: **incECNT**, **stEND**,
br step$_0$

! U korak $step_6$ se dolazi iz koraka $step_5$. U ovom koraku se bezuslovno generišu aktivne vrednost signala **incECNT** i **stEND** bloka *generisanje operacija* trajanja jedna perioda signala takta. Aktivnom vrednošću signala **incECNT** se inkrementira sadržaj brojača ECNT i time njegova vrednost podešava na prvi sledeći ulaz memorije TAB bloka *generisanje operacija*. Aktivnom vrednošću signala **stEND** treba da se u ovom zadnjem koraku generisanja operacije u slučaju kada je generisana operacija 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 $step_6$ se ostaje samo jadna perioda signala takta i bezuslovno se prelazi u korak $step_0$.

1.2.1.1.2.3. Vremenski oblici signala

Vremenski oblici signala dati su na slikama 10, 11 i 12. Slike 10 i 11 su date sa ciljem da se ilustruju vremenski oblici signala za sve četiri operacije i da se pokaže kako se zaustavlja generisanje operacija po generisanju operacije iz zadnjeg ulaza memorije TAB bloka *generisanje operacija*. Slika 12 je data sa ciljem da se pokaže kako se zaustavlja generisanje operacija kada sa u memoriji TAB naiđe na ulaz koji nije važeći.

Na slikama 10 i 11 je uzeto da su svi ulazi od 0 do 31 važeći. Dati su vremenski oblici signala za ulaze 0, 9, 18 i 31 za koje je uzeto da treba da se generišu operacije čitanja, upisa, selektivnog vraćanja i kompletnog vraćanja, respektivno.

Počinje se sa ulazom 0 za koji se generiše operacija čitanja. U koraku $step_0$ se, jer su signali **TV** i $\overline{\textbf{END}}$ bloka *generisanje operacija* aktivni, generiše aktivna vrednost signala **ldWCNT** bloka *generisanje operacija* i prelazi na korak $step_1$. U koraku $step_1$ se generiše aktivna vrednost signala **decWCNT** bloka *generisanje operacija* trajanja dve periode signala takta. To je zbog toga što je za operaciju čitanja uzeto da čekanje iznosi dva, pa su dve periode signala takta potrebne da sadržaj brojača WCNT dekrementiranjem padne na vrednost nula. Na isti signal takta na koji signal **decWCNT** postaje neaktivan, signal **OUT** bloka *generisanje operacija* postaje aktivan. Aktivna vrednost signala **OUT** u koraku $step_1$ se koristi da se generiše aktivna vrednost signala **ldMODE** bloka *keš interfejs* i da se na sledeći signal takta pređe na korak $step_2$. U koraku $step_2$ se ostaje samo jedna perioda signala takta. U njemu se, jer se radi o operaciji čitanja, generiše aktivna vrednost signala **ldPAR** bloka *keš interfejs*. Iz koraka $step_2$ se prelazi na korak $step_3$ i u njemu ostaje jedna perioda signala takta. U ovom koraku se generiše aktivna vrednost signala **PRQ** bloka *keš interfejs* i time, jer se radi o operaciji čitanja, dobija i aktivna vrednost signala **PRQRD**, kojom se startuje operacija čitanja u keš memoriji ***KEŠ***. Iz koraka $step_3$ se na signal takta uvek prelazi na korak $step_4$. U koraku $step_4$ se ostaje onoliko perioda signala takta koliko je potrebno keš memoriji ***KEŠ*** da kompletira operaciju čitanja. Pojava aktivne vrednosti signala **CRP** bloka *keš interfejs*, trajanja jedna perioda signala takta, je indikacija procesoru ***CPU*** da je operacija čitanja u keš memoriji ***KEŠ*** kompletirana. Na prvi sledeći signal takta posle onog kojim je signal **CRP** postao aktivan u registar $PDRDD_{7...0}$ bloka *keš interfejs* se upisuje očitana vrednost, jer je za operaciju čitanja signal **rd** aktivan, i prelazi iz koraka $step_4$ u korak $step_5$. U koraku $step_5$ se ostaje samo jedna perioda signala takta. U koraku $step_5$ se, jer se radi o operaciji čitanja, generiše aktivna vrednost signala **wrTAB** bloka *generisanje operacija* i na signal takta prelazi na korak $step_6$. U koraku $step_6$ se ostaje samo jedna perioda signala takta. U koraku $step_6$ se generišu aktivne vrednosti signala **incECNT** i **stEND** bloka *generisanje operacija*. Aktivnom vrednošću signala **incECNT** se na signal takta inkrementira sadržaj brojača ECNT i prelazi na ulaz 1 memorije TAB. Aktivna vrednost signala **stEND** ne utiče na flip-flop END, jer je signal **ecntones** neaktivan. Na signal takta se prelazi na korak $step_0$ generisanja sledeće operacije iz ulaza 1 memorije TAB.

Ulazi 1 do 8 memorije TAB se ne posmatraju. Sledeći ulaz koji se posmatra je ulaz 9 za koji se generiše operacija čitanja. Vremenski oblici signala koji se generišu u koracima $step_0$ do $step_7$ ulaza 9 su veoma slični vremenskim oblicima signala koji se generišu u koracima

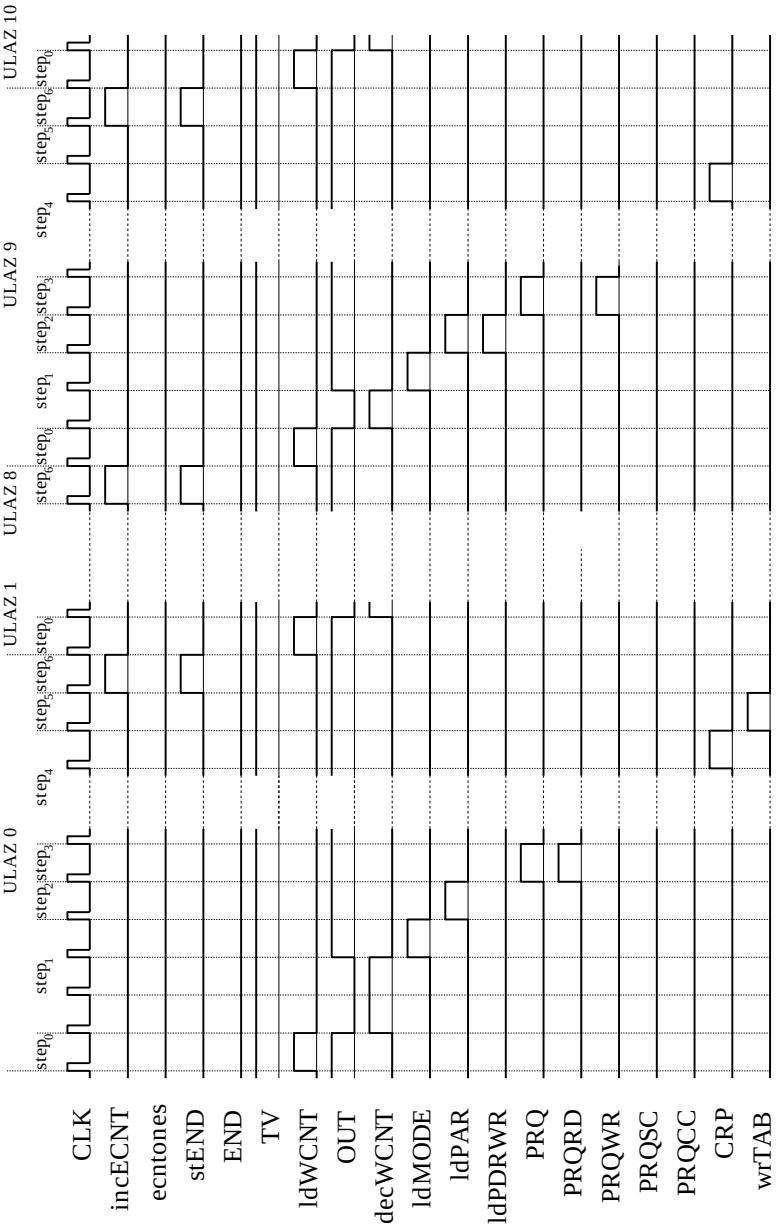
$step_0$ do $step_7$ ulaza 0. Zbog toga se ovde ukazuje samo na mesta na kojima se javljaju razlike. U koraku $step_1$ ulaza 9 se generiše aktivna vrednost signala **decWCNT** bloka *generisanje operacija* trajanja jedna perioda signala takta. To je zbog toga što je za operaciju upisa uzeto da čekanje iznosi jedan, pa je jedna perioda signala takta potrebna da sadržaj brojača WCNT dekrementiranjem padne na vrednost nula. U koraku $step_2$ ulaza 9 se, jer se radi o operaciji upisa, generiše aktivna vrednost signala **ldPDRWR** bloka *keš interfejs*. U koraku $step_3$ ulaza 9 se generiše aktivna vrednost signala **PRQ** bloka *keš interfejs* i time, jer se radi o operaciji upisa, dobija i aktivna vrednost signala **PRQWR**, kojom se startuje operacija upisa u keš memoriji ***KEŠ***. U koraku $step_4$ ulaza 9 se ostaje onoliko perioda signala takta koliko je keš memoriji ***KEŠ*** potrebno da kompletira operaciju upisa. Pojava aktivne vrednosti signala **CRP** bloka *keš interfejs*, trajanja jedna perioda signala takta, je indikacija procesoru ***CPU*** da je operacija upisa u keš memoriji ***KEŠ*** kompletirana, pa se na prvi sledeći signal takta posle onog kojim je signal **CRP** postao aktivan prelazi se iz koraka $step_4$ u korak $step_5$. U koraku $step_5$ ulaza 9 se ostaje samo jedna perioda signala takta. U koraku $step_5$ se, jer se radi o operaciji upisa, ne generiše aktivna vrednost signala **wrTAB** bloka *generisanje operacija* i na signal takta se prelazi na korak $step_6$. U koraku $step_6$ ulaza 9 se generišu aktivne vrednosti signala **incECNT** i **stEND** bloka *generisanje operacija* čime se na signal takta prelazi na korak $step_0$ generisanja sledeće operacije iz ulaza 10 memorije TAB.

Ulazi 10 do 17 memorije TAB se ne posmatraju. Sledeći ulaz koji se posmatra je ulaz 18 za koji se generiše operacija selektivnog vraćanja. Vremenski oblici signala koji se generišu u koracima $step_0$ do $step_7$ ulaza 9 su veoma slični vremenskim oblicima signala koji se generišu u koracima $step_0$ do $step_7$ ulaza 0. Zbog toga se ovde ukazuje samo na mesta na kojima se javljaju razlike. U koraku $step_1$ ulaza 18 se ne generiše aktivna vrednost signala **decWCNT** bloka *generisanje operacija*. To je zbog toga što je za operaciju selektivnog vraćanja uzeto da čekanje iznosi nula. Aktivnom vrednost signala **ldWCNT** bloka *generisanje operacija* u koraku $step_0$ ulaza 18 je na signal takta u brojač WCNT upisana nula, pa nema potrebe da se sadržaj brojača WCNT dekrementira. U koraku $step_3$ ulaza 9 se generiše aktivna vrednost signala **PRQ** bloka *keš interfejs* i time, jer se radi o operaciji selektivnog vraćanja, dobija i aktivna vrednost signala **PRQSC**, kojom se startuje operacija selektivnog vraćanja u keš memoriji ***KEŠ***. U koraku $step_4$ ulaza 9 se ostaje onoliko perioda signala takta koliko je keš memoriji ***KEŠ*** potrebno da kompletira operaciju selektivnog vraćanja. Pojava aktivne vrednosti signala **CRP** bloka *keš interfejs*, trajanja jedna perioda signala takta, je indikacija procesoru ***CPU*** da je operacija selektivnog vraćanja u keš memoriji ***KEŠ*** kompletirana, pa se na prvi sledeći signal takta posle onog kojim je signal **CRP** postao aktivan prelazi se iz koraka $step_4$ u korak $step_5$. U koraku $step_5$ ulaza 18 se ostaje samo jedna perioda signala takta. U koraku $step_5$ se, jer se radi o operaciji selektivnog vraćanja, ne generiše aktivna vrednost signala **wrTAB** bloka *generisanje operacija* i na signal takta se prelazi na korak $step_6$. U koraku $step_6$ ulaza 9 se generišu aktivne vrednosti signala **incECNT** i **stEND** bloka *generisanje operacija* čime se na signal takta prelazi na korak $step_0$ generisanja sledeće operacije iz ulaza 19 memorije TAB.

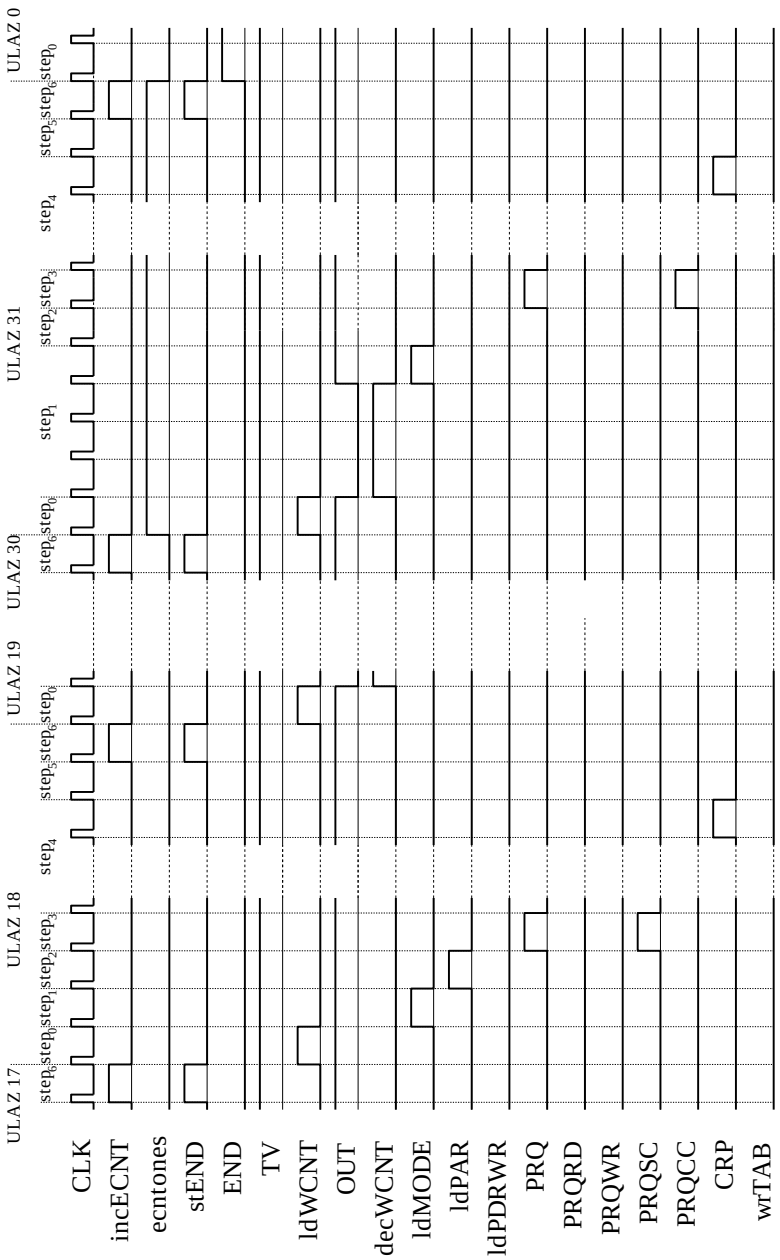
Ulazi 19 do 30 memorije TAB se ne posmatraju. Sledeći ulaz koji se posmatra je ulaz 31 za koji se generiše operacija kompletnog vraćanja. Vremenski oblici signala koji se generišu u koracima $step_0$ do $step_7$ ulaza 31 su veoma slični vremenskim oblicima signala koji se generišu u koracima $step_0$ do $step_7$ ulaza 0. Zbog toga se ovde ukazuje samo na mesta na kojima se javljaju razlike. U koraku $step_1$ ulaza 31 se generiše aktivna vrednost signala **decWCNT** bloka *generisanje operacija* trajanja tri periode signala takta. To je zbog toga što je za operaciju upisa uzeto da čekanje iznosi tri, pa su tri periode signala takta potrebne da sadržaj brojača WCNT dekrementiranjem padne na vrednost nula. U koraku $step_2$ ulaza 31 se, jer se radi o operaciji kompletnog vraćanja, ne generiše aktivna vrednost signala **ldPAR** bloka

keš interfejs. U koraku $step_3$ ulaza 9 se generiše aktivna vrednost signala **PRQ** bloka *keš interfejs* i time, jer se radi o operaciji kompletnog vraćanja, dobija i aktivna vrednost signala **PRQCC**, kojom se startuje operacija kompletnog vraćanja u keš memoriji ***KEŠ***. U koraku $step_4$ ulaza 9 se ostaje onoliko perioda signala takta koliko je keš memoriji ***KEŠ*** potrebno da kompletira operaciju kompletnog vraćanja. Pojava aktivne vrednosti signala **CRP** bloka *keš interfejs*, trajanja jedna perioda signala takta, je indikacija procesoru ***CPU*** da je operacija kompletnog vraćanja u keš memoriji ***KEŠ*** kompletirana, pa se na prvi sledeći signal takta posle onog kojim je signal **CRP** postao aktivan prelazi se iz koraka $step_4$ u korak $step_5$. U koraku $step_5$ ulaza 9 se ostaje samo jedna perioda signala takta. U koraku $step_5$ se, jer se radi o operaciji kompletnog vraćanja, ne generiše aktivna vrednost signala signala **wrTAB** bloka *generisanje operacija* i na signal takta se prelazi na korak $step_6$. U koraku $step_6$ se generišu aktivne vrednosti signala signala **incECNT** i **stEND** bloka *generisanje operacija*. **A**ktivnom vrednošću signala **incECNT** se na signal takta inkrementira sadržaj brojača ECNT i prelazi na ulaz 0 memorije TAB. Pošto je ovah zahtev generisan iz ulaza 31 memorije TAB, signal **ecntones** je aktivan. Zbog toga se aktivnom vrednošću signala **stEND** na signal takta flip-flop END postavlja na aktivnu vrednost. U koraku $step_6$ se ostaje samo jedna perioda signala takta, pa se na signal takta prelazi na korak $step_0$ generisanja sledeće operacije iz ulaza 0 memorije TAB. Međutim, u koraku $step_0$ ulaza 0 je sada aktivna vrednost signala **END** bloka *generisanje operacija*. Zbog toga sada nema generisanja aktivne vrednosti signala **ldWCNT** bloka *generisanje operacija* i nema prelasaka iz koraka $step_0$ u korak $step_1$, već se ostaje u koraku $step_0$. Tek kada se generiše aktivna vrednost signala opšteg brisanja **mr** i ceo sistem dovede u početno stanje, signal **END** postaje neaktivan, pa se ponovo kreće sa generisanjem operacija počev od ulaza 0 memorije TAB ukoliko je bit TV ulaza 0 jedinica.

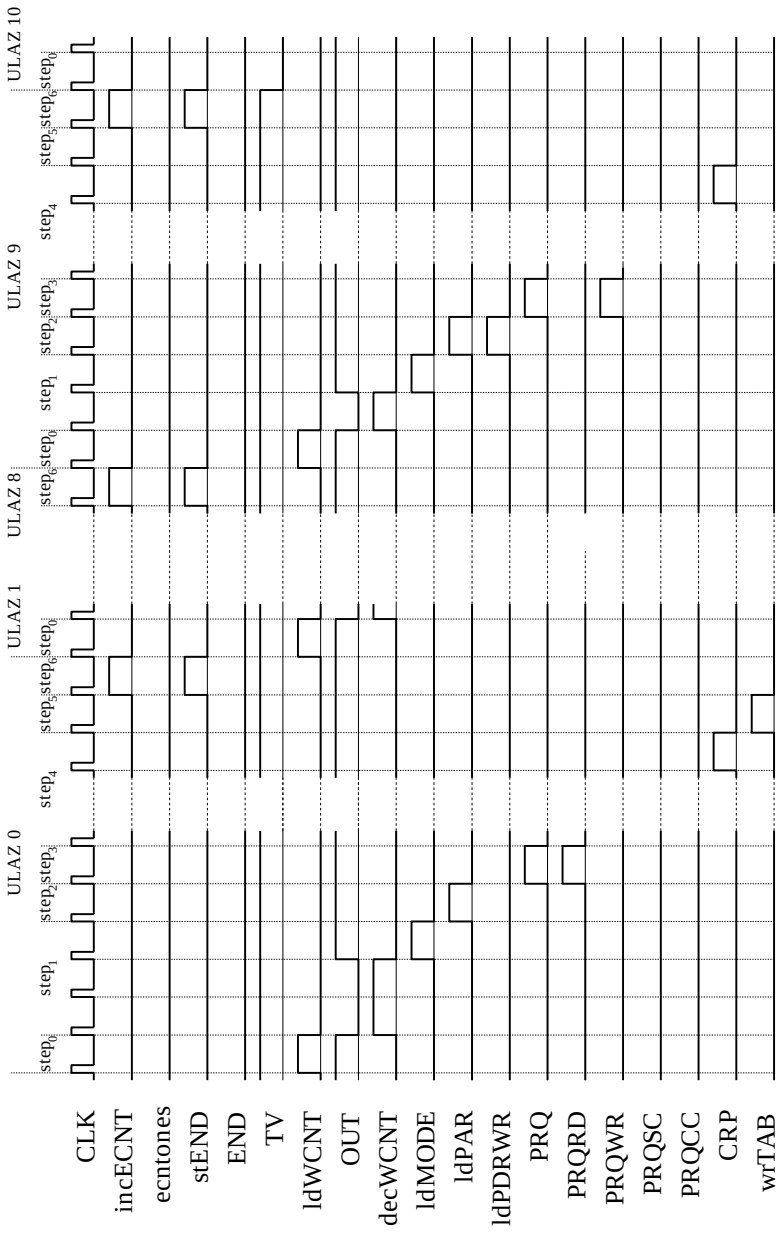
Na slici 12 je uzeto da su ulazi od 0 do 9 važeći, a da ulaz 10 nije važeći. Uzeto je da se za ulaze 0 i 9 generišu operacije čitanja i upisa sa identičnim karakteristikama kao i za operacije čitanja i upisa za ulaze 0 i 9 sa slike 10 iz prethodnog primera. Stoga su sva objašnjenja za sliku 12 identična sa objašnjenjima sa slike 10 zaključno sa ulazom 9. Na signal takta u koraku $step_6$ ulaza 9 brojač ECNT bloka *generisanje operacija* se inkrementira na 10 i prelazi na $step_0$ ulaza 10. Kako je za ovaj ulaz bit TV nula, nema generisanja aktivne vrednosti signala **ldWCNT** bloka *generisanje operacija* i nema prelaska iz koraka $step_0$ u korak $step_1$, već se ostaje u koraku $step_0$. Tek kada se generiše aktivna vrednost signala opšteg brisanja **mr** i ceo sistem dovede u početno stanje, sadržaj brojača ECNT bloka *generisanje operacija* postaje nula, signal **END** bloka *generisanje operacija* postaje neaktivan, pa se ponovo kreće sa generisanjem operacija počev od ulaza 0 memorije TAB ukoliko je bit TV ulaza 0 jedinica.



Slika 10 Vremenski oblici signala u procesoru za kompletno popunjenu memoriju TAB (prvi deo)



Slika 11 Vremenski oblici signala u procesoru za kompletno popunjenu memoriju TAB (drugi deo)



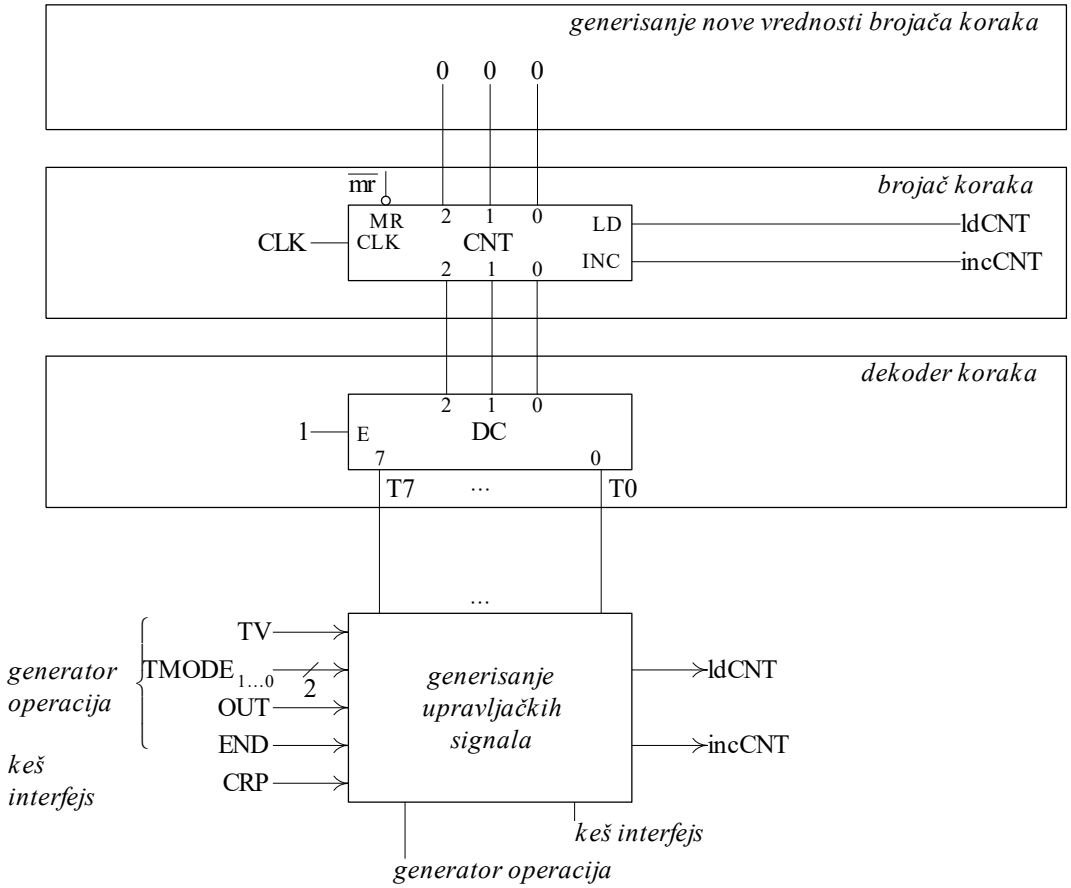
Slika 12 Vremenski oblici signala u procesoru za delimično popunjenu memoriju TAB

1.2.1.1.2.4. Struktura upravljačke jedinice

Struktura ***upravljačke jedinice*** ožičene realizacije je prikazana na slici 13. ***Upravljačka jedinica*** 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.



Slika 13 Struktura upravljačke jedinice

1.2.1.1.2.4.1. Blok generisanje nove vrednosti brojača koraka

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 1.2.1.1.2.2) se utvrđuje da je 0 vrednost koju treba upisati u brojač CNT. Ta vrednost je ožičena na ulazima 0, 1 i 2 brojača CNT.

1.2.1.1.2.4.2. Blok brojač koraka

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 1.2.1.1.2.2). 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 1.2.1.1.2.2). 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**.

1.2.1.1.2.4.3. Blok dekoder koraka

Blok *dekoder koraka* sadrži dekoder DC. Na ulaze dekodera DC vode se izlazi brojača CNT. Dekodovana stanja brojača CNT pojavljuju se kao signali **T0** do **T7** na izlazima dekodera DC. Svakom koraku iz algoritma generisanja upravljačkih signala (poglavlje 1.2.1.1.2.2) dodeljen je jedan od ovih signala i to koraku $step_0$ signal **T0**, koraku $step_1$ signal **T1**, itd.

1.2.1.1.2.4.4. Blok generisanje upravljačkih signala

Blok *generisanje upravljačkih signala* sadrži kombinacione mreže koje pomoću signala **T0** do **T7** koji dolaze sa bloka *dekoder koraka* ***upravljačke jedinice***, signala logičkih uslova koji dolaze iz blokova ***operacione jedinice*** i saglasno algoritmu generisanja upravljačkih signala (poglavlje 1.2.1.1.2.2) 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 **ldMODE** treba da je aktivan u koraku $step_1$ ukoliko je signal logičkog uslova **OUT** aktivan, pa se ovaj upravljački signal generiše prema relaciji **ldMODE** = **T1** · **OUT**, dok upravljački signal **PRQ** treba da je uvek aktivan u koraku $step_3$, pa se ovaj upravljački signal generiše prema relaciji **PRQ** = **T3**.

Blok *generisanje upravljačkih signala* generiše dve grupe upravljačkih signala i to:
- upravljačke signale operacione jedinice i
- upravljačke signale upravljačke jedinice.

1.2.1.1.2.4.4.1. Upravljački signali operacione jedinice

Upravljački signali operacione jedinice podeljeni su u grupe koje odgovaraju blokovima ***operacione jedinice*** i to:
- blok keš interfejs i
- blok generator operacija.

1.2.1.1.2.4.4.1.1. Blok keš interfejs

Upravljački signali koji odgovaraju ovom bloku ***operacione jedinice*** su:
- **ldPDRWR** — signal paralelnog upisa u registar PDRWR,
- **ldMODE** — signal paralelnog upisa u registar MODE,
- **ldPAR** — signal paralelnog upisa u registar PAR i

- **PRQ** — signal startovanja jedne od četiri operacije keš memorije ***KEŠ***.

Ovi upravljački signali se generišu na sledeći način:

- **ldPDRWR** = **T2** · ($\overline{\mathbf{TMODE_1}}$ · **TMODE$_0$**),
- **ldMODE** = **T1** · **OUT**,
- **ldPAR** = **T2** · $\overline{(\mathbf{TMODE_1} \cdot \mathbf{TMODE_1})}$ i
- **PRQ** = **T3**.

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

- **TMODE$_{1...0}$** — blok generisanje operacija i
- **OUT** — blok generisanje operacija.

1.2.1.1.2.4.4.1.2. Blok generisanje operacija

Upravljački signali koji odgovaraju ovom bloku ***operacione jedinice*** su:

- **ldWCNT** — signal paralelnog upisa u brojač WCNT,
- **decWCNT** — signal dekrementiranja sadržaja brojača WCNT,
- **incECNT** — signal inkrementiranja sadržaja brojača ECNT,
- **wrTAB** — signal upisa sadržaja u memoriju TAB i
- **stEND** — signal upisa aktivne vrednosti u flip-flopa END.

Ovi upravljački signali se generišu na sledeći način:

- **ldWCNT** = **T0** · **TV** · $\overline{\mathbf{END}}$,
- **decWCNT** = **T1** · $\overline{\mathbf{OUT}}$,
- **incECNT** = **T6**,
- **wrTAB** = **T5** · ($\overline{\mathbf{TMODE_1}}$ · $\overline{\mathbf{TMODE_0}}$) i
- **stEND** = **T6**.

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

- **TV** — blok generisanje operacija,
- **END** — blok generisanje operacija,
- **OUT** — blok generisanje operacija i
- **TMODE$_{1...0}$** — blok generisanje operacija.

1.2.1.1.2.4.4.2. Upravljački signali upravljačke jedinice

Upravljački signali upravljačke jedinice su:

- **ldCNT** — signal paralelnog upisa u brojač CNT i
- **incCNT** — signal inkrementiranja sadržaja brojača CNT.

Ovi upravljački signali se generišu na sledeći način:

- **ldCNT** = **T6** i
- **incCNT** = **T0** · **TV** · $\overline{\mathbf{END}}$ + **T1** · **OUT** + **T2** + **T3** + **T4** · **CRP** + **T5**.

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

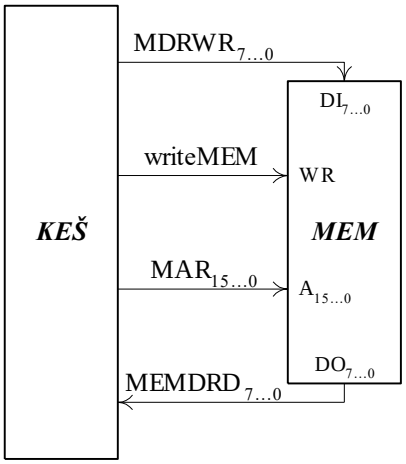
- **TV** — blok generisanje operacija,
- **END** — blok generisanje operacija,

- **OUT** — blok generisanje operacija i
- **CRP** — blok keš interfejs.

1.2.1.2. MEMORIJA MEM

Memoriji ***MEM*** se obraća keš memorija ***KEŠ*** ili kod prebacivanja bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM*** kada treba realizovati upise u memoriju ***MEM*** ili kod prebacivanja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** kada treba realizovati čitanja iz memorije ***MEM*** (slika 14).

Kod čitanja iz memorije ***MEM*** keš memorija ***KEŠ*** šalje 16-bitnu adresu po linijama $\mathbf{MAR_{15...0}}$ i drži neaktivnu vrednost signala **writeMEM**. Očitani 8-bitni podatak se vraća po linijama $\mathbf{MEMDRD_{7...0}}$. Kod upisa u memoriju ***MEM*** keš memorija ***KEŠ*** šalje 16-bitnu adresu po linijama $\mathbf{MAR_{15...0}}$, 8-bitni podatak po linijama $\mathbf{MDRWR_{7...0}}$ i drži aktivnu vrednost signala **writeMEM**. Uzeto je da je vreme pristupa memorije ***MEM*** jednako četiri periode signala takta.



Slika 14 Memorija MEM

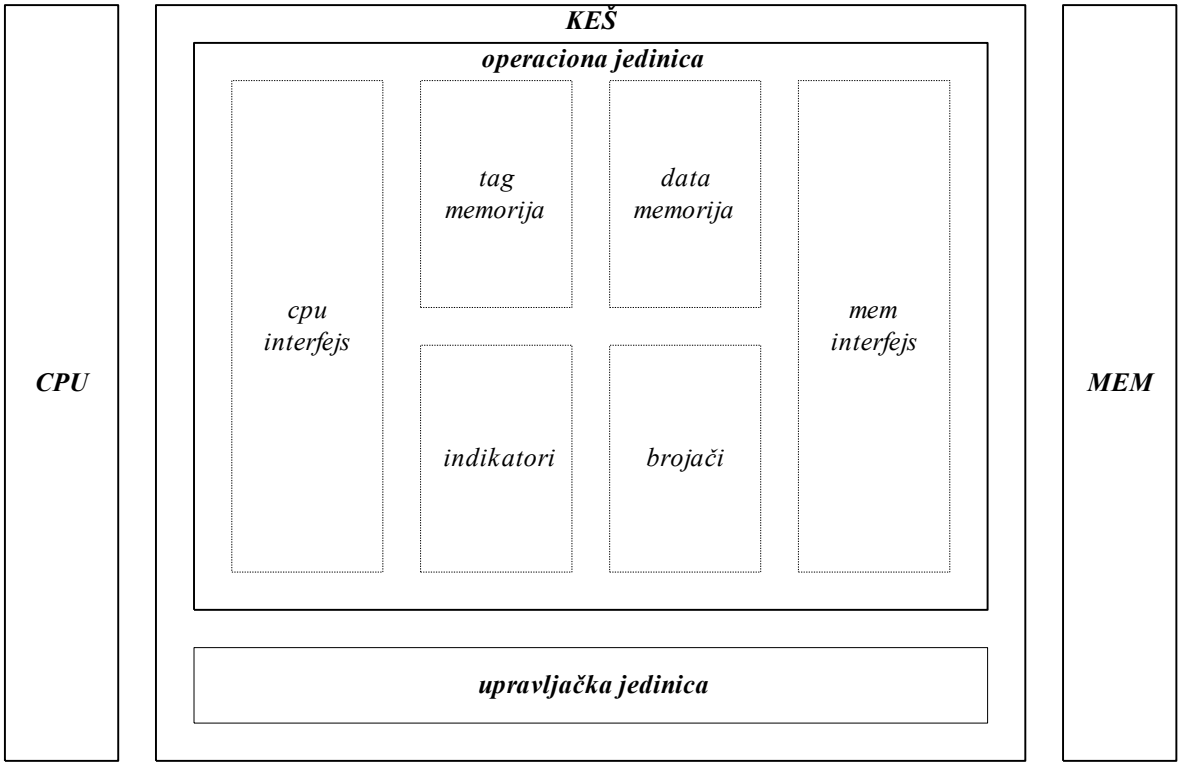
1.2.1.3. KEŠ MEMORIJA KEŠ

Keš memorija ***KEŠ*** (slika 15) 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 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 prema algoritmu generisanja upravljačkih signala operacija keš memorije KEŠ i signala logičkih uslova.

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



Slika 15 Keš memorija KEŠ

1.2.1.3.1. Operaciona jedinica

Operaciona jedinica (slika 15) se sastoji iz sledećih blokova:

- blok *cpu interfejs*,
- blok *indikatori*,
- blok *brojači*,
- blok *tag memorija*,
- blok *data memorija* i
- blok *mem interfejs*.

Blok *cpu interfejs* služi za razmenu adresa, podataka i upravljačkih signala između keš memorije ***KEŠ*** i procesora ***CPU***.

Blok *indikatori* služi za vođenje evidencije za svaki od 8 ulaza keš memorije o tome da li je ulaz važeći i da li je sadržaj bloka koji se nalazi u ulazu modifikovan nekim prethodnim upisom.

Blok *brojači* služi za generisanje adresa bajtova unutar bloka prilikom prebacivanja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** obratno, za prebrojavanje prenetih bajtova iz memorije ***MEM*** u keš memoriju ***KEŠ*** i obratno, za odbrojavanje onoliko perioda signala takta koliko je vreme pristupa memoriji ***MEM***, kao i za generisanje broja ulaza u keš memoriju ***KEŠ***.

Blok *tag memorija* služi za vođenje evidencije za svaki od 8 ulaza keš memorije ***KEŠ*** o tome kojoj grupi pripada blok podataka koji se nalazi u datom ulazu keš memorije ***KEŠ***.

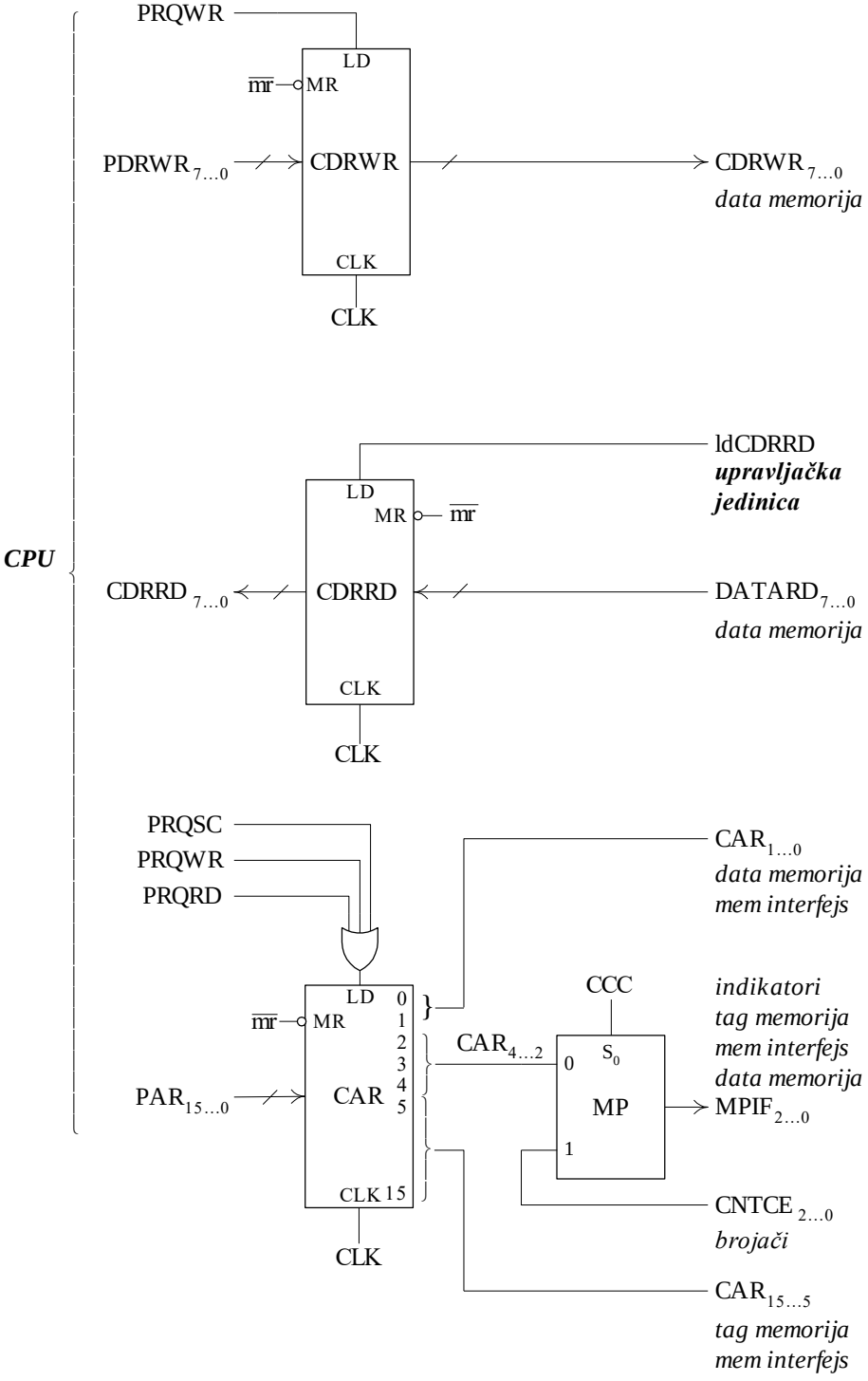
Blok *data memorija* služi za smeštanje 8 blokova podataka.

Blok *mem interfejs* služi za razmenu adresa, podataka i upravljačkih signala između keš memorije ***KEŠ*** i memorije ***MEM***.

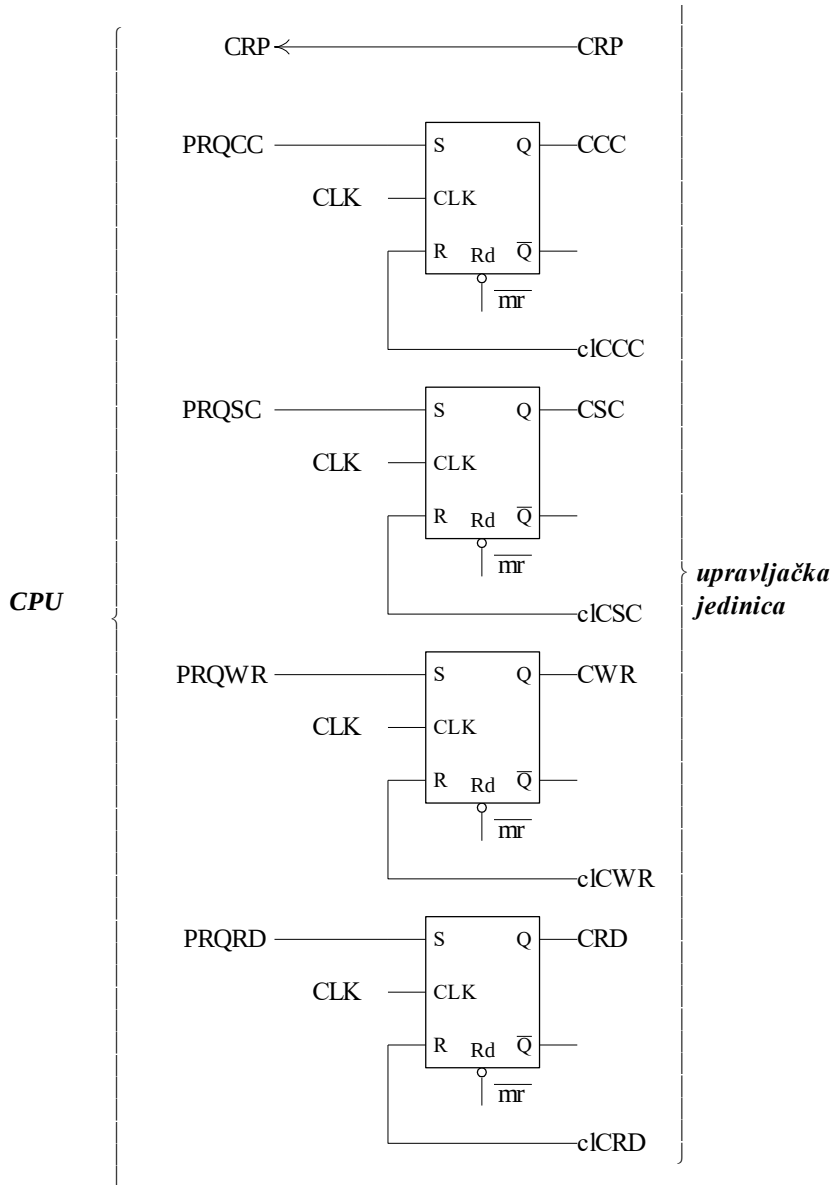
Struktura i opis blokova ***operacione jedinice*** daju se u daljem tekstu.

1.2.1.3.1.1. Blok cpu interfejs

Blok *cpu interfejs* (slike 16 i 17) sadrži registre CAR, CDRWR i CDRRD, multiplekser MP i flip-flopove CRD, CWR, CSC i CCC.



Slika 16 Blok cpu interfejs (prvi deo)



Slika 17 Blok cpu interfejs (drugi deo)

Registar CAR (Cache Address Register) služi za čuvanje ili 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 ili adrese lokacije memorije ***MEM*** koja pripada bloku koji treba, ukoliko je modifikovan, vratiti iz keš memorije ***KEŠ*** u memoriju ***MEM*** u slučaju operacije selektivnog vraćanja. Na ulaze registra CAR dovodi se 16-bitna adresa iz procesora ***CPU*** po linijama $\mathbf{PAR_{15...0}}$. Ova adresa se upisuje u registar CAR pri pojavi signala takta ukoliko je aktivan jedan od signala **PRQRD, PRQWR** ili **PRQSC** koji dolaze od procesora ***CPU***.

Signali $\mathbf{CAR_{15...5}}$ sa izlaza registra CAR se vode:

- u blok *tag memorija* gde se koriste za utvrđivanja saglasnosti kod operacija čitanja, upisa i selektivnog vraćanja, i za upisivanje u memorijski modul TAG kod dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** i
- u blok *mem interfejs* gde se koriste u formiranju adrese memorije ***MEM*** kod dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** i kod vraćanja bloka podataka iz keš memoriju ***KEŠ*** u memoriju ***MEM***.

Signali **$CAR_{4...2}$** sa izlaza registra CAR se vode na ulaz multipleksera MP.

Multiplekser MP služi za selekciju signala **$CAR_{4...2}$** i **$CNTCE_{2...0}$** i formiranje signala **$MPIF_{2...0}$**. U slučaju operacija čitanja, upisa ili selektivnog vraćanja signal **CCC** je neaktivan, pa se kroz multiplekser propuštaju signali **$CAR_{4...2}$**. U slučaju operacije kompletnog vraćanja signal **CCC** je aktivan, pa se kroz multiplekser propuštaju signali **$CNTCE_{2...0}$**. Selektovani signali se kao signali **$MPIF_{2...0}$** vode u :

- blok *tag memorija* radi adresiranja ulaza memorijskog modula TAG i blok *indikatori* radi adresiranja indikatora V i indikatora D,
- u blok *data memorija* gde se koriste u formiranju adrese memorijskog modula DATA sa koje treba ili nešto očitati ili na kojoj treba nešto upisati i
- u blok *mem interfejs* gde se koriste u formiranju adresa memorije ***MEM*** kod dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** i kod vraćanja bloka podataka iz keš memoriju ***KEŠ*** u memoriju ***MEM***.

Registar CDRWR (Cache Data Register for WRite) služi za čuvanje podatka koji treba upisati u slučaju operacije upisa. Na ulaze registra CDRWR dovodi se po linijama **$PDRWR_{7...0}$** 8-bitni podatak iz procesora ***CPU***. Ovaj podatak se upisuje u registar CDRWR pri pojavi signala takta ukoliko je aktivan signal **PRQWR** koji dolazi od procesora ***CPU***. Sadržaj registra CDRWR se vodi u blok *data memorija* u kome se upisuje u memorijski modul DATA u slučaju operacije upisa.

Registar CDRRD (Cache Data Register for ReaD) služi za čuvanje očitanog podatka u slučaju operacije čitanja. Na ulaze registra CDRRD dovodi se po linijama **$DATARD_{7...0}$** 8-bitni podatak iz memorijskog modula DATA iz bloka *data memorija*. Ovaj podatak se upisuje u registar CDRRD pri pojavi signala takta ukoliko je aktivan signal **ldCDRRD** koji generiše ***upravljačka jedinica***.

Aktivnom vrednošću signala **PRQRD, PRQWR, PRQSC** ili **PRQCC** trajanja jedne periode signala takta procesor ***CPU*** signalizira keš memoriji ***KEŠ*** da treba da uradi operaciju čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja, respektivno. Aktivnom vrednošću signala **CRP** trajanja jedne periode signala takta keš memorija ***KEŠ*** signalizira procesoru ***CPU*** da je odgovarajuća operacija izvršena.

Flip-flopovi CRD (Cache ReaD), CWR (Cache WRite), CSC (Cache Selectively Clear) i CCC (Cache Completely Clear) služe da se u keš memoriji ***KEŠ*** upamti da su u toku operacije čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja, respektivno. Flip-flop CRD se postavlja na aktivnu vrednost na signal takta pri pojavi aktivne vrednosti signala **PRQRD** koji dolazi iz procesora ***CPU*** i traje jednu periodu signala takta. Aktivna vrednost ostaje sve dok je operacija čitanja u toku. Flip-flop CRD se postavlja na neaktivnu vrednost na kraju operacije čitanja na signal takta pri aktivnoj vrednosti signala **clCRD** trajanja jedne periode signala takta. Za flip-flopove CWR, CSC i CCC važi sve što je rečeno za flip-flop CRD. Tako se u slučaju operacije upisa flip-flop CWR postavlja na aktivnu vrednost pri pojavi aktivne vrednosti signala **PRQWR** i postavlja na neaktivnu vrednost pri pojavi aktivne vrednosti signala **clCWR**. U slučaju operacije selektivnog vraćanja flip-flop CSC se postavlja na aktivnu vrednost pri pojavi aktivne vrednosti signala **PRQSC** i postavlja na neaktivnu vrednost pri pojavi aktivne vrednosti signala **clCSC**. U slučaju operacije kompletnog vraćanja flip-flop CCC se postavlja na aktivnu vrednost pri pojavi aktivne vrednosti signala **PRQCC** i postavlja na neaktivnu vrednost pri pojavi aktivne vrednosti signala **clCCC**.

1.2.1.3.1.2. Blok indikatori

Blok *indikatori* (slika 18) sadrži flip-flopove V_0 do V_7 i D_0 do D_7, dekoder DC i multiplekser MP.

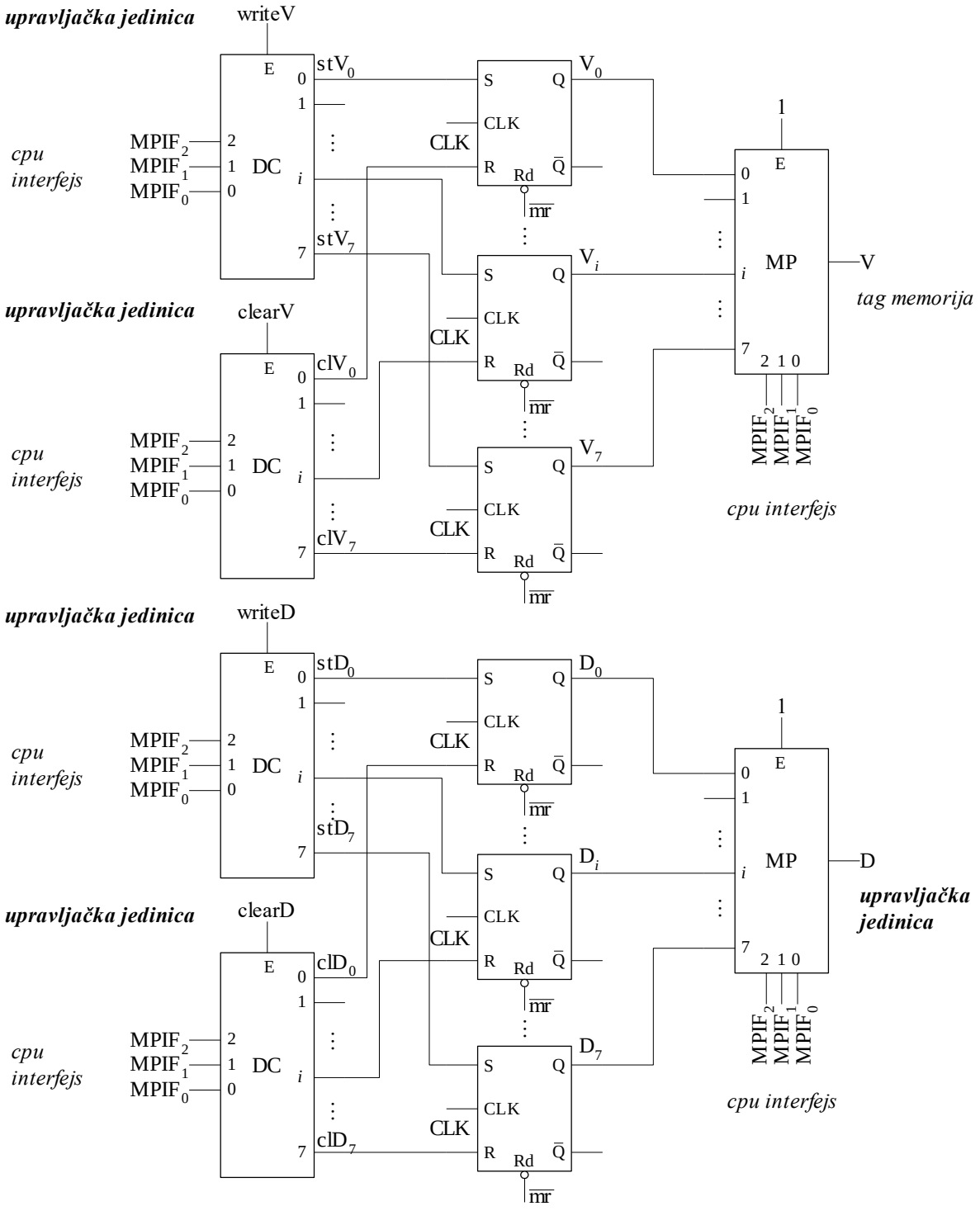
Flip-flopovima V_0 do V_7 služe za čuvanje indikatora V (Valid). Njima se za svaki ulaz keš memorije ***KEŠ*** vodi evidencija da li je važeći ili ne. U slučaju operacija čitanja i upisa se prilikom dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** generiše aktivna vrednost signala **writeV**. Tada se na signal takta upisuje aktivna vrednost u jedan od flip-flopova V_0 do V_7 i to onaj koji adresiran signalima $\mathbf{MPIF_{2...0}}$ iz bloka *cpu interfejs*. Signali $\mathbf{MPIF_{2...0}}$ su tada formirani od signala $\mathbf{CAR_{4...2}}$ generisane adrese (odeljak 1.2.1.3.1.1). U slučaju operacija selektivnog vraćanja i kompletnog vraćanja se prilikom vraćanja bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM*** generiše aktivna vrednost signala **clearV**. Tada se na signal takta upisuje neaktivna vrednost u jedan od flip-flopova V_0 do V_7 i to onaj koji adresiran signalima $\mathbf{MPIF_{2...0}}$. Signali $\mathbf{MPIF_{2...0}}$ su formirani od signala $\mathbf{CAR_{4...2}}$ generisane adrese u slučaju operacije selektivnog vraćanja i signala $\mathbf{CNTE_{2...0}}$ brojača CNTCE u slučaju operacije kompletnog vraćanja (odeljak 1.2.1.3.1.1).

Dekoderi DC uz flip-flopove V_0 do V_7 služe za formiranje signala $\mathbf{stV_0}$ do $\mathbf{stV_7}$ i $\mathbf{clV_0}$ do $\mathbf{clV_7}$. Pri aktivnoj vrednosti signala **writeV**, a u zavisnosti od vrednosti signala $\mathbf{MPIF_{2...0}}$, generiše se aktivna vrednost jednog od signala $\mathbf{stV_0}$ do $\mathbf{stV_7}$ i time određuje jedan od flip-flopova V_0 do V_7 u koji se upisuje aktivna vrednost. Pri aktivnoj vrednosti signala **clearV**, a u zavisnosti od vrednosti signala $\mathbf{MPIF_{2...0}}$, generiše se aktivna vrednost jednog od signala $\mathbf{clV_0}$ do $\mathbf{clV_7}$ i time određuje jedan od flip-flopova V_0 do V_7 u koji se upisuje neaktivna vrednost.

Multiplekser MP uz flip-flopove V_0 do V_7 služi za formiranje signala **V**. Ovaj signal u slučaju operacija čitanja, upisa, selektivnog vraćanja i kompletnog vraćanja predstavlja indikaciju da li je određeni ulaz keš memorije ***KEŠ*** važeći. To je ulaz keš memorije ***KEŠ*** određen signalima $\mathbf{CAR_{4...2}}$ generisane adrese u slučaju operacija čitanja, upisa i selektivnog vraćanja i signalima $\mathbf{CNTCE_{2...0}}$ brojača CNTCE u slučaju operacije kompletnog vraćanja. Na informacione ulaze multipleksera MP vode se signali $\mathbf{V_0}$ do $\mathbf{V_7}$, a na upravljačke ulaze signali $\mathbf{MPIF_{2...0}}$. Signali $\mathbf{MPIF_{2...0}}$ su formirani od signala $\mathbf{CAR_{4...2}}$ generisane adrese u slučaju operacija čitanja, upisa i selektivnog vraćanja i signala $\mathbf{CNTCE_{2...0}}$ brojača CNTCE u slučaju operacije kompletnog vraćanja (odeljak 1.2.1.3.1.1).

Flip-flopovima D_0 do D_7 služe za čuvanje indikatora D (Dirty). Njima se za svaki ulaz keš memorije ***KEŠ*** vodi evidencija da li je bilo upisa u blok datog ulaza ili ne. Ako je bilo upisa, a radi se o operaciji čitanja ili upisa, tada se pre dovlačenja novog bloka iz memorije ***MEM*** u dati ulaz keš memorije ***KEŠ*** blok iz datog ulaza keš memorije ***KEŠ*** vraća u memoriju ***MEM***. Ako nije bilo upisa, odmah se prelazi na dovlačenje novog bloka iz memorije ***MEM*** u dati ulaz keš memorije ***KEŠ***. Ako je bilo upisa, a radi se o operaciji selektivnog vraćanja ili kompletnog vraćanja, tada se blok iz datog ulaza keš memorije ***KEŠ*** vraća u memoriju ***MEM***. Ako nije bilo upisa, a radi se o operaciji selektivnog vraćanja, operacija se završava. Ako nije bilo upisa, a radi se o operaciji kompletnog vraćanja, prelazi se na istu vrstu provere za prvi sledeći ulaz keš memorije ***KEŠ***. Ukoliko se u slučaju operacije upisa otkrije saglasnost i upis izvrši u keš memoriju ***KEŠ***, generiše se aktivna vrednost signala **writeD**. Tada se na signal takta upisuje aktivna vrednost u jedan od flip-flopova D_0 do D_7 i to onaj koji adresiran signalima $\mathbf{MPIF_{2...0}}$. Signali $\mathbf{MPIF_{2...0}}$ su formirani od signala $\mathbf{CAR_{4...2}}$ generisane adrese (odeljak 1.2.1.3.1.1). U slučaju operacija selektivnog vraćanja i kompletnog vraćanja se prilikom vraćanja bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM*** generiše aktivna vrednost signala **clearV**. Tada se na signal takta upisuje neaktivna vrednost u jedan od flip-flopova D_0 do D_7 i to onaj koji adresiran signalima $\mathbf{MPIF_{2...0}}$. Signali $\mathbf{MPIF_{2...0}}$ su formirani

od signala **CAR$_{4...2}$** generisane adrese u slučaju operacije selektivnog vraćanja i signala **CNTCE$_{2...0}$** brojača CNTCE u slučaju operacije kompletnog vraćanja (odeljak 1.2.1.3.1.1).



Slika 18 Blok indikatori

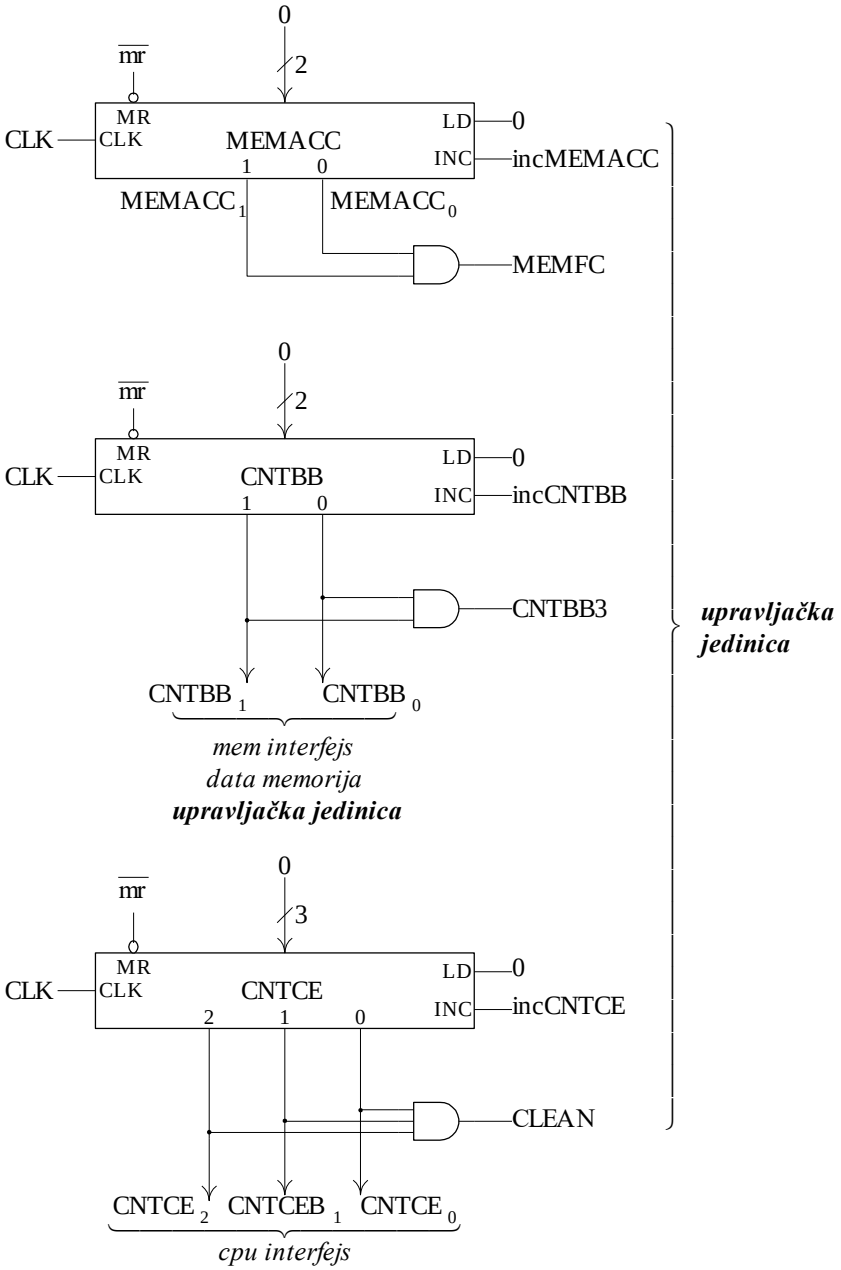
Dekoderi DC uz flip-flopove D_0 do D_7 služe za formiranje signala **stD$_0$** do **stD$_7$** i **clD$_0$** do **clD$_7$**. Pri aktivnoj vrednosti signala **writeD**, a u zavisnosti od vrednosti signala **MPIF$_{2...0}$**, generiše se aktivna vrednost jednog od signala **stD$_0$** do **stD$_7$** i time određuje jedan od flip-flopova D_0 do D_7 u koji se upisuje aktivna vrednost. Pri aktivnoj vrednosti signala **clearD**, a u

zavisnosti od vrednosti signala $\mathbf{MPIF_{2...0}}$, generiše se aktivna vrednost jednog od signala $\mathbf{clD_0}$ do $\mathbf{clD_7}$ i time određuje jedan od flip-flopova D_0 do D_7 u koji se upisuje neaktivna vrednost.

Multiplekser MP uz flip-flopove D_0 do D_7 služi za formiranje signala **D**. Ovaj signal u slučaju operacija čitanja, upisa, selektivnog vraćanja i kompletnog vraćanja predstavlja indikaciju da li je određeni ulaz keš memorije ***KEŠ*** modifikovan. To je ulaz keš memorije ***KEŠ*** određen signalima $\mathbf{CAR_{4...2}}$ generisane adrese u slučaju operacija čitanja, upisa i selektivnog vraćanja i signalima $\mathbf{CNTCE_{2...0}}$ brojača CNTCE u slučaju operacije kompletnog vraćanja. Na informacione ulaze multipleksera MP vode se signali $\mathbf{D_0}$ do $\mathbf{D_7}$, a na upravljačke ulaze signali $\mathbf{MPIF_{2...0.}}$ Signali $\mathbf{MPIF_{2...0}}$ su formirani od signala $\mathbf{CAR_{4...2}}$ generisane adrese u slučaju operacija čitanja, upisa i selektivnog vraćanja i signala $\mathbf{CNTCE_{2...0}}$ brojača CNTCE u slučaju operacije kompletnog vraćanja (odeljak 1.2.1.3.1.1).

1.2.1.3.1.3. Blok brojači

Blok *brojači* (slika 19) sadrži brojač bajtova bloka CNTBB (Counter of Bytes of Blocks), brojača vremena pristupa operativnoj memoriji MEMACC (MEMory ACCess time) i brojača ulaza u keš memoriju CNTCE (CouNTer of Cach Entry).



Slika 19 Blok brojači

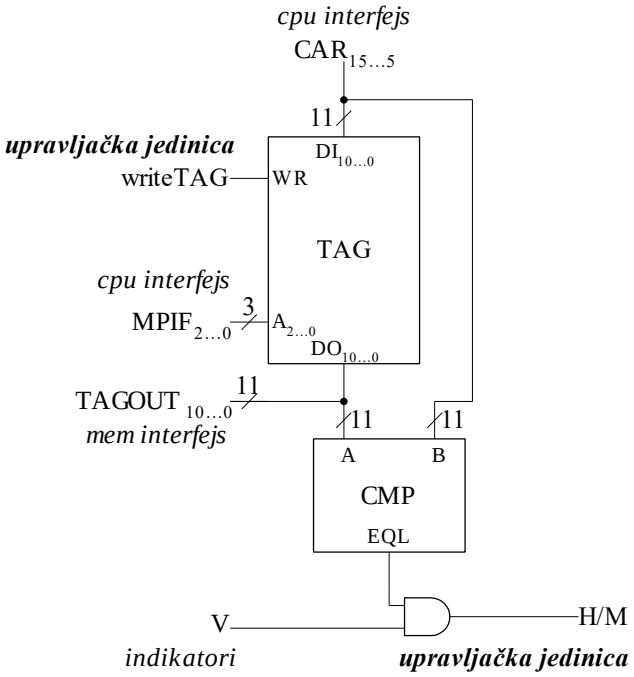
Brojač CNTBB se koristi za generisanje adresa bajtova u bloku kod dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** i kod vraćanja bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***. Brojač CNTBB se koristi u bloku *data memorija* kod formiranja adrese memorijskog modula DATA i u bloku *mem interfejs* kod formiranja adresa memorije ***MEM***.

Brojač MEMACC se koristi da odbroji četiri signala takta kod obraćanja keš memorije ***MEM*** memoriji ***MEM*** radi čitanja ili upisa, jer je usvojeno da je vreme pristupa memoriji ***MEM*** četiri periode signala takta. Signal **MEMFC** (MEMory Function Completed) postaje aktivan kada brojač MEMACC pređe u stanje tri. Na četvrti signal takta brojač MEMACC se vraća na stanje nula, a signal **MEMFC** postaje neaktivan. Signal **MEMFC** služi ***upravljačkoj jedinici*** kao indikacija da je pristup memoriji ***MEM*** završen.

Brojač CNTCE se koristi kod operacije kompletnog vraćanja. Sadržaj brojača CNTCE predstavlja ulaz keš memorije ***KEŠ*** iz koga se blok podataka prebacuje u memoriju ***MEM***. Kada svi izlazi brojača CNTCE postanu aktivni, što odgovara zadnjem ulazu keš memorije ***KEŠ***, generiše se aktivna vrednost signala **CLEAN**. Ovaj signal služi ***upravljačkoj jedinici*** kao indikacija da je završeno vraćanje svih modifikovanih blokova iz keš memorije ***KEŠ*** u memoriju ***MEM***.

1.2.1.3.1.4. Blok tag memorija

Blok *tag memorija* (slika 20) sadrži RAM memorijski modul TAG i komparator CMP.



Slika 20 Blok tag memorija

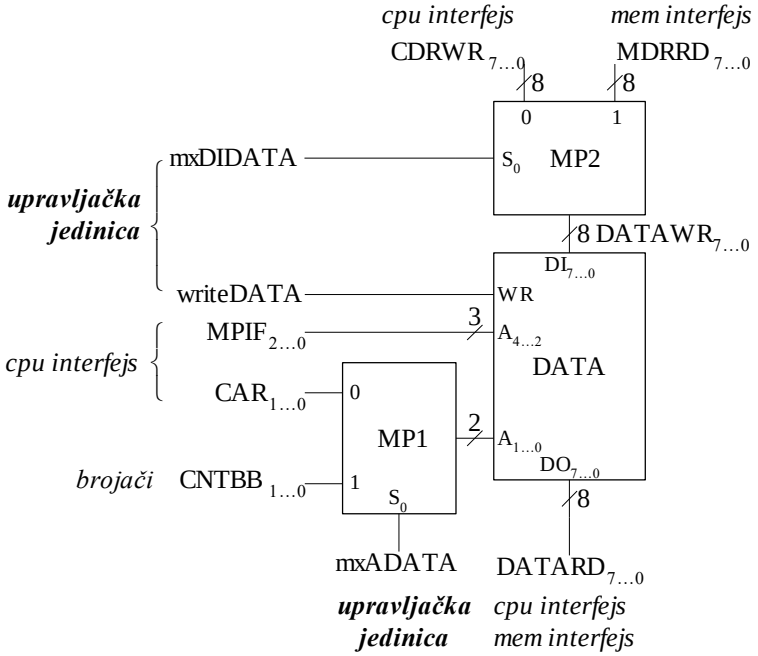
Memorijski modul TAG čuva za svaki od 8 ulaza keš memorije ***KEŠ*** broj bloka memorije ***MEM*** koji se trenutno nalazi u keš memoriji ***KEŠ***. Kapacitet modula TAG je 8 reči širine 11 bita, a adresiranje se vrši signalima $\mathbf{MPIF_{2..0}}$ koji se vode na ulazne linije $A_{2...0}$ memorijskog modula TAG. U slučaju operacija čitanja, upisa i selektivnog vraćanja signali $\mathbf{MPIF_{2...0}}$ se formiraju od signala $\mathbf{CAR_{4...2}}$ generisane adrese. Ovi signali predstavljaju broj bloka u grupi memorije ***MEM*** i broj ulaza u keš memoriji ***KEŠ***. U slučaju operacije kompletnog vraćanja signali $\mathbf{MPIF_{2...0}}$ se formiraju od signala $\mathbf{CNTCE_{2...0}}$ brojača CNTCE. Sadržaj brojača CNTCE predstavlja ulaz keš memorije ***KEŠ*** iz koga se blok podataka ukoliko je modifikovan prebacuje u memoriju ***MEM***. Očitana vrednost se pojavljuje kao signali $\mathbf{TAGOUT_{10...0}}$ na izlaznim linijama podataka $DO_{10...0}$ memorijskog modula TAG. Ona predstavlja broj grupe bloka memorije ***MEM*** koji se nalazi u ulazu memorijskog modula TAG adresiranom signalima $\mathbf{MPIF_{2...0}}$. Očitana vrednost se vodi na jedan od ulaza komparatora CMP radi utvrđivanja saglasnosti i u blok *mem interfejs* radi formiranja adresa memorije ***MEM*** kod vraćanja bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***. Signali $\mathbf{CAR_{15...5}}$ koji se vode na ulazne linije podataka $DI_{10...0}$ memorijskog modula TAG predstavljaju broj grupe bloka memorije ***MEM*** koji treba upisati u adresirani ulaz memorijskog modula TAG kod dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ***. Ta ista vrednost se vodi na drugi ulaz komparatora CMP radi utvrđivanja saglasnosti. Upis u memorijski modul TAG

se vrši generisanjem aktivne vrednosti signala **writeTAG**, a čitanje neaktivnom vrednošću tog signala.

Komparator CMP upoređuje bitove $CAR_{15...5}$ broja grupe generisane adrese i bitove $TAGOUT_{10...0}$ očitanog broja grupe iz memorijskog modula TAG. Ako su ove vrednosti iste i ako je pri tome signal **V** je aktivan, što znači da je sadržaj u adresiranom ulazu keš memorije ***KEŠ*** važeći, dobija se aktivna vrednost signala **H/M**. U ovom slučaju se kaže da postoji saglasnost.

1.2.1.3.1.5. Blok data memorija

Blok *data memorija* (slika 21) sadrži memorijski modul DATA i multiplekser MP1 i MP2.



Slika 21 Blok data memorija

Memorijski modul DATA čuva sadržaje 8 blokova podataka koji se trenutno nalaze u keš memoriji ***KEŠ***. Ovaj modul ima 32 lokacije širine jedan bajt organizovanih u 8 blokova po 4 bajta. S toga je adresa memorijskog modula DATA širine 5 bita, gde niža 2 bita definišu bajt unutar bloka, a viša 3 bita određuju broj bloka keš memorije ***KEŠ***.

Memorijski modul DATA ima sledeće ulaze i izlaze:

- adresne linije $A_{4...0}$,
- ulazne linije podataka $DI_{7...0}$,
- zlazne linije podataka $DO_{7...0}$ i
- upravljačku liniju 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:

1. *Kod operacije čitanja pri čitanju bajta podatka iz keš memorije **KEŠ** ako je otkrivena saglasnost*. U ovom slučaju signali na adresnim linijama $A_{4...0}$ memorijskog modula DATA se formiraju od signala $\mathbf{CAR_{4...2}}$ i $\mathbf{CAR_{1...0}}$ sa izlaza registra CAR bloka *cpu interfejs*. Signali $\mathbf{CAR_{4...2}}$ se propuštaju kroz multiplekser sa slike 16 i pojavljuju kao signali $\mathbf{MPIF_{2...0}}$ na adresnim linijama $A_{4...2}$, dok se signali $\mathbf{CAR_{1...0}}$ propuštaju kroz

multiplekser sa slike 21 i pojavljuju na adresnim linijama $A_{1...0}$. Očitani podatak se pojavljuje kao signali **$DATARD_{7...0}$** na izlaznim linijama podataka $DO_{7...0}$ memorijskog modula DATA i vodi se u blok *cpu interfejs*.

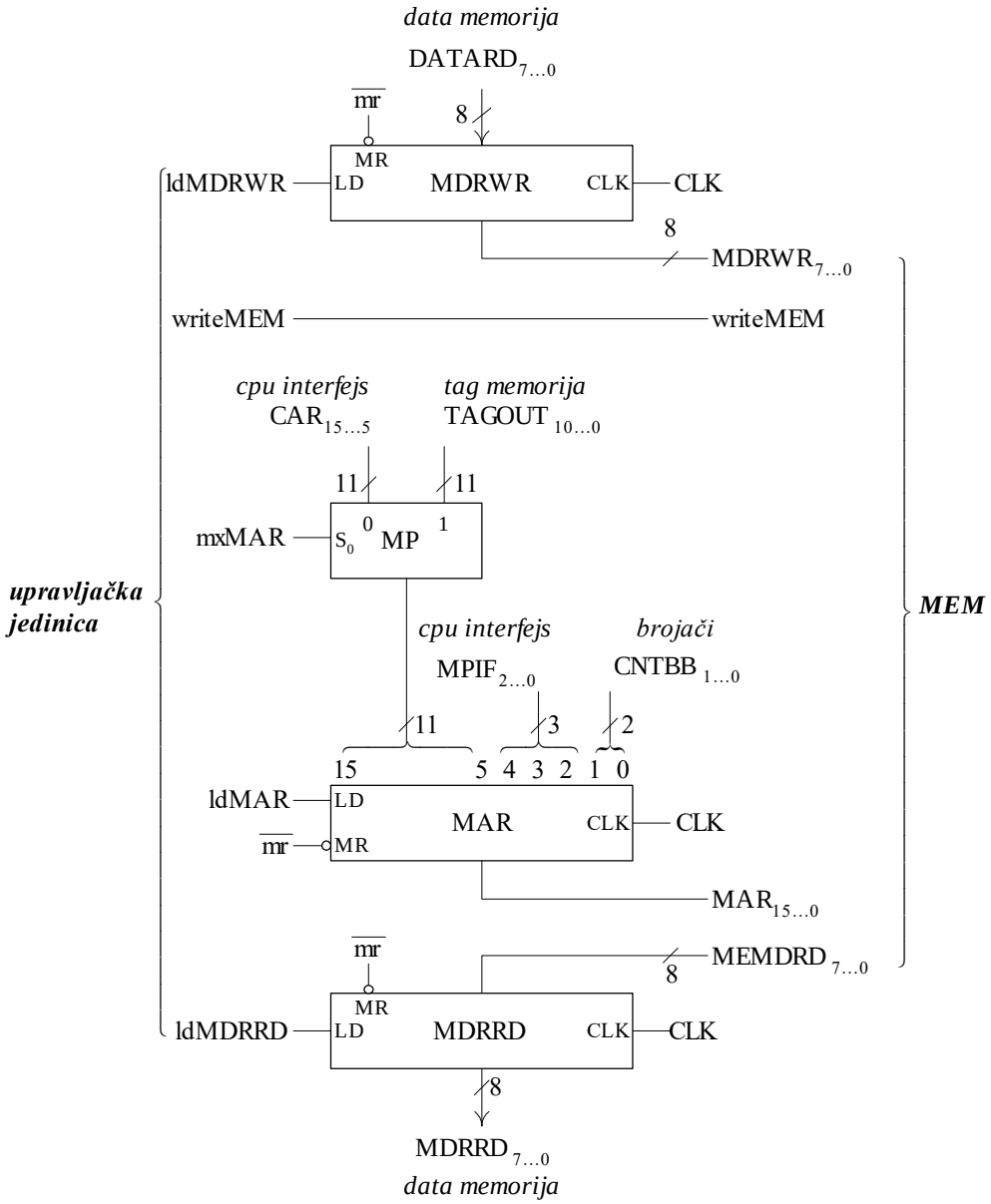
2. *Kod operacije upisa pri upisu bajta podatka u keš memoriju **KEŠ** ako je otkrivena saglasnost*. Adresa se formira kao u slučaju 1. Podatak za upis u memorijski modul DATA se formira od signala **$CDRWR_{7...0}$** iz bloka *cpu interfejs*. Signali **$CDRWR_{7...0}$** se propuštaju kroz multiplekser i pojavljuju kao signali **$DATAWR_{7...0}$** na ulaznim linijama podataka $DI_{7...0}$ memorijskog modula DATA.
3. *Kod operacija čitanja i upisa pri čitanju iz keš memorije **KEŠ** bajtova bloka podataka koji se vraćaju iz keš memorije **KEŠ** u memoriju MEM ako nema saglasnosti a sadržaj bloka adresiranog ulaza keš memorije **KEŠ** je modifikovan*. U ovom slučaju signali na adresnim linijama $A_{4...0}$ memorijskog modula DATA se formiraju od signala **$CAR_{4...2}$** i **$CNTBB_{1...0}$** sa izlaza registra CAR bloka *cpu interfejs* i brojača $CNTBB_{1...0}$ bloka *brojači*, respektivno. Signali **$CAR_{4...2}$** se propuštaju kroz multiplekser sa slike 16 i pojavljuju kao signali **$MPIF_{2...0}$** na adresnim linijama $A_{4...2}$, dok se signali **$CNTBB_{1...0}$** propuštaju kroz multiplekser sa slike 21 i pojavljuju na adresnim linijama $A_{1...0}$. Očitani podatak se pojavljuje kao signali **$DATARD_{7...0}$** na izlaznim linijama podataka $DO_{7...0}$ memorijskog modula DATA i vodi u blok *mem interfejs*.
4. *Kod operacija čitanja i upisa pri upisu u keš memoriju **KEŠ** bajtova bloka podataka koji se dovlače iz memorije **MEM** u keš memoriju **KEŠ** ako nema saglasnosti*. Adresa bajta u memorijskom modulu DATA se formira kao u slučaju 3. Podatak za upis u memorijski modul DATA se formira od signala **$MDRRD_{7...0}$** iz bloka *cpu interfejs*. Signali **$MDRRD_{7...0}$** se propuštaju kroz multiplekser i pojavljuju kao signali **$DATAWR_{7...0}$** na ulaznim linijama podataka $DI_{7...0}$ memorijskog modula DATA.
5. *Kod operacije selektivnog vraćanja pri čitanju bajtova bloka koji se vraćaju iz keš memorije **KEŠ** u memoriju **MEM** ako ima saglasnosti i ako je sadržaj bloka adresiranog ulaza keš memorije **KEŠ** modifikovan*. Formiranje adrese i čitanje iz memorijskog modula DATA se realizuje kao u slučaju 3.
6. *Kod operacije kompletnog vraćanja pri čitanju bajtova bloka podataka koji se vraćaju iz keš memorije **KEŠ** u memoriju **MEM** ako ima saglasnosti i ako je sadržaj bloka adresiranog ulaza keš memorije **KEŠ** modifikovan*. U ovom slučaju signali na adresnim linijama $A_{4...0}$ memorijskog modula DATA se formiraju od signala **$CNTCE_{2...0}$** i **$CNTBB_{1...0}$** sa izlaza brojača CNTCE i CNTBB bloka *brojači*. Signali **$CNTCE_{2...0}$** se propuštaju kroz multiplekser sa slike 16 i pojavljuju kao signali **$MPIF_{2...0}$** na adresnim linijama $A_{4...0}$, dok se signali **$CNTBB_{1...0}$** propuštaju kroz multiplekser sa slike 21 i pojavljuju na adresnim linijama $A_{1...0}$. Očitani podatak se pojavljuje kao signali **$DATARD_{7...0}$** na izlaznim linijama podataka $DO_{7...0}$ memorijskog modula DATA i vodi se u blok *mem interfejs*.

Signali na adresnim linijama $A_{4...2}$ i $A_{1...0}$ memorijskog modula DATA formirani su na sledeći način. Na adresne linije $A_{4...2}$ vode sa signali **$MPIF_{2...0}$** sa izlaza multipleksera sa slike 16. Na adresne linije $A_{1...0}$ vode se signali sa izlaza multipleksera sa slike 21. U slučajevima 1 do 5 radi se operacijama čitanja, upisa i selektivnog vraćanja, pa je upravljački signal **CCC** multipleksera sa slike 16 neaktivan. Stoga se kao signali **$MPIF_{2...0}$** pojavljuju signali **$CAR_{4...2}$** . U slučaju 6 radi se o operaciji kompletnog vraćanja, pa je upravljački signal **CCC** multipleksera sa slike 16 neaktivan. Stoga se kao signali **$MPIF_{2...0}$** pojavljuju signali **$CNTCE_{2...0}$** . U slučajevima 1 i 2 radi se o čitanju i upisu samo jednog bajta, pa se neaktivnom vrednošću signala **mxDATA** kroz multiplekser MP1 sa slike 21 na adresna linije $A_{1...0}$ propuštaju signali **$CAR_{1...0}$** registra CAR. U slučajevima 3 do 6 radi se o čitanju ili upisu bloka bajtova, pa se aktivnom vrednošću signala **mxADATA** kroz multiplekser sa slike 21 na adresne linije $A_{1...0}$ propuštaju signali **$CNTBB_{1...0}$** sa izlaza brojača CNTBB modula *brojači*.

Signali na ulaznim linijama podataka $DI_{7...0}$ memorijskog modula DATA su signali **$DATAWR_{7...0}$** koji se vode sa izlaza multipleksera sa slike 21. U slučaju 2 radi se o upisu bajta podatka iz procesora ***CPU*** u keš memoriju ***KEŠ***, pa se neaktivnom vrednošću signala **mxDIDATA** kroz multiplekser MP2 propuštaju signali **$CDRWR_{7...0}$** registra CDRWR iz bloka *keš interfejs*. U slučaju 4 radi se o upisu bajtova bloka podataka koji se dovlači iz memorije ***MEM*** u keš memoriju ***KEŠ***, pa se aktivnom vrednošću signala **mxDIDATA** kroz multiplekser propuštaju signali **$MDRWR_{7...0}$** registra MDRWR iz bloka *mem interfejs*.

1.2.1.3.1.6. Blok mem interfejs

Blok *mem interfejs* (slika 22) sadrži registre MAR, MDRWR i MDRRD i multiplekser MP.



Slika 22 Blok mem interfejs

Registar MAR (Memory Address Register) služi za čuvanje ili adrese lokacije memorije ***MEM*** sa koje treba očitati podatak u slučaju dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** ili adrese lokacije memorije ***MEM*** u koju treba upisati podatak u slučaju

vraćanja bloka podataka iz keš memorije ***MEM*** u memoriju ***MEM***. Adresa se upisuje u registar MAR pri pojavi signala takta i aktivnoj vrednosti signala **ldMAR**. Adresa koja se upisuje formira se iz tri dela i to na sledeći način:

1. *Bitovi 15 do 5*. Kao bitovi 15 do 5 se koriste ili signali $\mathbf{CAR_{15...5}}$ registra CAR iz bloka *cpu interfejs* ili signali $\mathbf{TAGOUT_{10...0}}$ sa izlaza memorijskog modula TAG iz bloka *tag memori*ja. Signali $\mathbf{CAR_{15...5}}$ se koriste kod dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ***, a signali $\mathbf{TAGOUT_{10...0}}$ kod vraćanja bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***. Neaktivnom i aktivnom vrednošću upravljačkog signala **mxMAR** kroz multiplekser se selektuju signali $\mathbf{CAR_{15...5}}$ i $\mathbf{TAGOUT_{10...0}}$, respektivno.
2. *Bitovi 4 do 2*. Kao bitovi 4 do 2 se koriste signali $\mathbf{MPIF_{2..0}}$ sa izlaza multipleksera iz bloka *cpu interfejs* (slika 16). Pri operacijama čitanja, upisa i selektivnog vraćanja kao signali $\mathbf{MPIF_{2..0}}$ pojavljuju se signali $\mathbf{CAR_{4..2}}$ registra CAR iz bloka *cpu interfejs*, dok se pri operaciji kompletnog vraćanja kao signali $\mathbf{MPIF_{2..0}}$ pojavljuju signali $\mathbf{CNTCE}_{2..0}$ brojača CNTCE iz bloka *brojači*.
3. *Bitovi 1 do 0*. Kao bitovi 1 do 0 se koriste signali $\mathbf{CNTBB_{1...0}}$ brojača CNTBB iz bloka *brojači*.

Izlazi registra MAR se kao signali $\mathbf{MAR_{15..0}}$ vode na adresne linije $A_{15..0}$ memorije ***MEM***.

Registar MDRWR (Memory Data Register for WRite) služi za čuvanje podataka koje treba upisati u memoriju ***MEM*** pri vraćanju bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***. Podatak za upis se dovodi na ulaze registra MDRWR kao signali $\mathbf{DATARD_{7...0}}$ iz bloka *data memorija*. Ovaj podatak se upisuje u registar MDRWR na signal takta ukoliko je aktivan signal **ldMDRWR**. Izlazi registra MDRWR se kao signali $\mathbf{MDRWR_{7..0}}$ vode na ulazne linije podataka $DI_{7..0}$ memorije ***MEM***.

Registar MDRRD (Memory Data Register for ReaD) služi za čuvanje podatka koji je očitan iz memorije ***MEM*** pri dovlačenju bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ***. Ovaj podatak se kao signali $\mathbf{MEMDRD_{7...0}}$ dovodi sa izlaznih linija podataka $DO_{7...0}$ memorije ***MEM*** na ulaze registra MDRRD i upisuje u registar MDRRD na signal takta ukoliko je aktivan signal **ldMDRRD**. Izlazi registra MDRRD se kao signali $\mathbf{MDRRD_{7...0}}$ vode u blok *data memorija* radi upisa u memorijski modul DATA.

1.2.1.3.2. Upravljačka jedinica

Upravljačka jedinica keš memorije ***KEŠ*** služi za generisanje upravljačkih signala prema algoritmu direktnog peslikavanja. U ovom poglavlju se daju dijagrami toka operacija, algoritam generisanja upravljačkih signala, vremenski oblici signala i struktura ***upravljačke jedinice***.

1.2.1.3.2.1. Dijagram toka operacija

Dijagram toka operacija je dat na slikama 23 i 24.

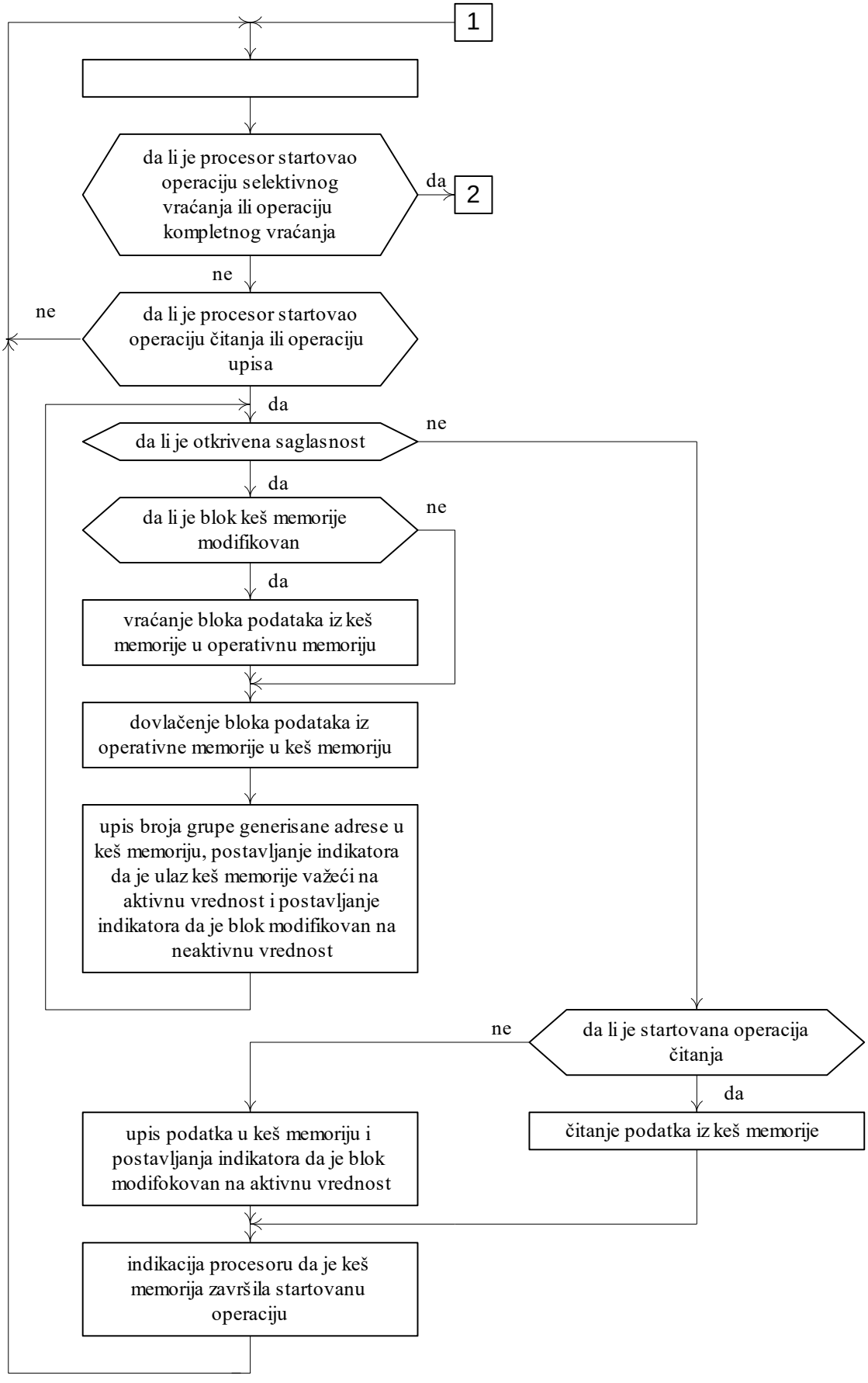
U početnom stanju upravljačka jedinica keš memorije ***KEŠ*** čeka pojavu signala startovanja operacije čitanja **PRQRD,** signala startovanja operacije upisa **PRQWR,** signala startovanja operacije selektivnog vraćanja **PRQSC** i signala startovanja operacije kompletnog vraćanja **PRQCC**. U slučaju operacija čitanja, upisa i selektivnog vraćanja adresa operativne memorije koju šalje procesor se upisuje u prihvatni registar adrese keš memorije ***KEŠ***, dok se u slučaju operacija upisa podatak koji šalje procesor upisuje u prihvatni registar podatka keš memorije ***KEŠ***.

Sledeći korak u slučaju operacija čitanja**,** upisa i selektivnog vraćanja je provera da li se sadržaj sa date adrese nalazi u keš memoriji ***KEŠ***. Ova provera će se dalje nazivati provera da li postoji saglasnost. Ukoliko postoji saglasnost vrši se čitanje podatka iz keš memorije ***KEŠ*** i upis u prihvatni registar podatka keš memorije ***KEŠ*** u slučaju operacije čitanja, odnosno upis podatka iz prihvatnog registra podatka keš memorije ***KEŠ*** u keš memoriju ***KEŠ*** i postavljanje indikatora o modifikaciji bloka keš memorije ***KEŠ*** u koji je izvršen upis u slučaju operacije upisa. Na kraju se šalje signal **CRP** koji predstavlja indikaciju o završenom čitanju iz keš memorije ***KEŠ*** ili upisu u keš memoriju ***KEŠ***. U slučaju opracije čitanja očitani podatak se prebacuje iz prihvatnog registra podatka keš memorije ***KEŠ*** u prihvatni registar podatka procesora ***CPU***. Po slanju signala **CRP** se prelazi na čekanje signala startovanja nove operacije.

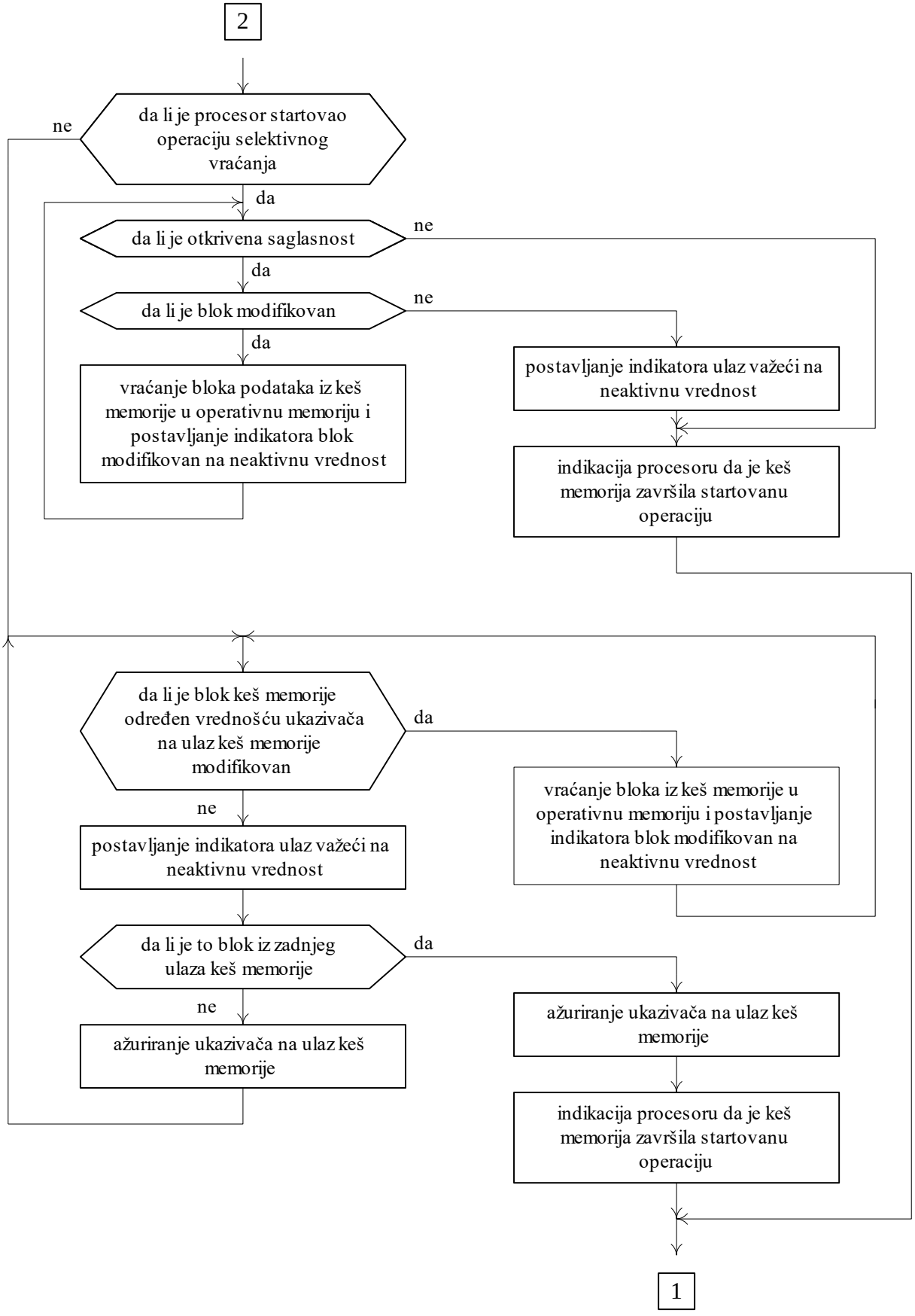
Ukoliko ne postoji saglasnost prelazi se na dovlačenje bloka iz memorije ***MEM*** u keš memoriju ***KEŠ***. Ovome može da prethodi vraćanje bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM*** ukoliko je sadržaj bloka iz ulaza u koji treba dovući novi blok modifikovan. Ovo je posledica odluke da se za ažuriranje memorije ***MEM*** koristi metoda *vrati-nazad*.

Potom se prelazi na već opisani korak u kome se vrši provera da li se sadržaj sa date adrese nalazi u keš memoriji ***KEŠ***. Pošto je potreban blok dovučen iz memorije ***MEM*** u keš memoriju ***KEŠ***, sada će postojati saglasnost, pa će se dalji rad odvijati na već opisani način za slučaj saglasnosti.

Keš memorija ***KEŠ*** obrađuje operacije selektivno vraćanje tako što ide od prvog do poslednjeg ulazu u keš memoriji ***KEŠ*** i proverava da li je blok koji se u njemu nalazi modifikovan ili nije (ukoliko jeste, postavljen je D bit). Ukoliko je blok modifikovan, vraća se u memoriju ***MEM***, a D i V bitovi koji odgovaraju tom ulazu resetuju se. Ukoliko blok nije modifikovan, nema potrebe za ažuriranjem sadržaja memorije ***MEM***, dovoljno je samo resetovati V bit odgovarajućeg ulaza (D bit je već na nuli - to je indikacija da blok nije modifikovan).



Slika 23 Dijagram toka operacija (prvi deo)



Slika 24 Dijagram toka operacija (drugi deo)

1.2.1.3.2.2. Algoritam generisanja upravljačkih signala

Algoritam generisanja upravljačkih signala je formiran na osnovu strukture ***operacione jedinice*** (poglavlje 1.2.1.3.1) i dijagrama toka operacija (poglavlje 1.2.1.3.2.1). Notacija koja se koristi je identična sa notacijom iz poglavlja 1.2.1.1.2.2.

Algoritam generisanja upravljačkih signala je dat u daljem tekstu.

$step_0$: *br* (*if* (**PRQRD** + **PRQWR** + **PRQSC** + **PRQCC**) *then* $step_1$ *else* $step_0$)

! U koraku $step_0$ se čeka da iz procesora ***CPU*** preko bloka *cpu interfejs* stigne signal startovanja operacije čitanja **PRQRD,** signal startovanja operacije upisa **PRQWR**, signal startovanja operacije selektivnog vraćanja **PRQSC** ili signal startovanja operacije kompletnog vraćanja **PRQCC**. Ako se pojavi signal **PRQRD** na signal takta se adresa upisuje u registar CAR bloka *cpu interfejs* i flip-flop CRD bloka *cpu interfejs* postavlja na aktivnu vrednost. Ako se pojavi signal **PRQWR** na signal takta se adresa upisuje u registar CAR, podatak u registar CDRWR bloka *cpu interfejs* i flip-flop CWR bloka *cpu interfejs* postavlja na aktivnu vrednost. Ako se pojavi signal **PRQSC** na signal takta se adresa upisuje u registar CAR i flip-flop CSC bloka *cpu interfejs* postavlja na aktivnu vrednost. Ako se pojavi signal **PRQCC** na signal takta flip-flop CCC bloka *cpu interfejs* se postavlja na aktivnu vrednost. Ako se pojavi bilo koji od signala **PRQRD, PRQWR, PRQSCL** ili **PRQCC** na signal takta se prelazi na korak $step_1$. U suprotnom slučaju se ostaje u koraku $step_0$.

$step_1$: *if* ((**CRD** · **H/M**), **ldCDRRD**),
if ((**CSC** · **H/M** ·), c**learV**),
if ((**CCC** ·), **incCNTCE, clearV**),
br (*if* (**CRD** + **CWR**)
then (*if* **H/M** *then* $step_2$ *else* (*if* **D** *then* $step_3$ *else* $step_5$))
else (*if* **CSC**
then (*if* **H/M**
then (*if* **D** *then* $step_3$ *else* $step_2$)
else $step_2$)
else (*if* **D**
then $step_3$
else (*if* **CLEAN** *then* $step_2$ *else* $step_1$))))

! U korak $step_1$ može da se dođe iz koraka $step_0$, iz koraka $step_7$ i iz koraka $step_4$. Iz koraka $step_0$ se dolazi uvek kada se startuje operacija čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja. Iz koraka $step_7$ se dolazi u slučaju operacija čitanja i upisa po izvršenom dovlačenju bloka padataka iz memorije ***MEM*** u keš memoriju ***KEŠ***. Iz koraka $step_4$ se dolazi u slučaju operacija selektivnog vraćanja ili kompletnog vraćanja po izvršenom prebacivanju bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***.

U koraku $step_1$ se za operacije čitanja, upisa i selektivnog vraćanja vrši provera saglasnosti i na osnovu toga formira vrednost signala **H/M** bloka *tag memorija*. Aktivna vrednost signala **H/M** označava da je otkrivena saglasnost. U koraku $step_1$ se za sve četiri operacije pod određenim uslovima vrši provera da li sadržaj bloka podataka određenog ulaza keš memorije modifikovan i na osnovu toga formira vrednost signala **D** bloka *indikatori*. Aktivna vrednost signala **D** označava da je sadržaj modifikovan. U koraku $step_1$ se za operaciju kompletnog vraćanja pod određenim uslovima vrši i provera vrednosti signala **CLEAN** bloka *brojači*. Aktivna vrednost signala **CLEAN** je indikacija da su svi modifikovan blokovi keš memorije ***KEŠ*** vraćeni u memoriju ***MEM***. Signal **CLEAN** postaje aktivan kada brojač CNTCE bloka *brojači* inkrementiranjem od početne vrednosti nula dostigne krajnju vrednost sedam.

U slučaju operacije čitanja signal **CRD** je aktivan. Korak $step_1$ služi za formiranje signala **H/M**. Ako je formirana aktivna vrednost signala **H/M** generiše se aktivna vrednost **ldCDRRD** bloka *cpu interfejs*, pa se na signal takta vrednost očitana iz memorijskog modula DATA upisuje u registar CDRRD bloka *cpu interfejs*. U slučaju operacije upisa signal **CWR** je aktivan. Korak $step_1$ služi samo za formiranje signala **H/M**. U slučaju operacija čitanja i upisa iz koraka $step_1$ se prelazi na korak $step_2$, $step_3$ ili $step_5$ u zavisnosti od vrednosti signala **H/M** i **D**. Ako je signal **H/M** aktivan, prelazi se na korak $step_2$ radi kompletiranja date operacije. Ako je signal **H/M** neaktivan, prelazi se u zavisnosti od vrednosti signala **D** na korak $step_3$ ili korak $step_5$. Pri aktivnoj vrednosti signala **D**, što znači da je sadržaj bloka modifikovan, prelazi se na korak $step_3$. U koracima $step_3$ i $step_4$ se vraćaju bajtovi bloka iz keš memorije ***KEŠ*** u memoriju ***MEM***. Pri neaktivnoj vrednosti signala **D**, što znači da sadržaj bloka nije modifikovan, prelazi se na korak $step_5$. U koracima $step_5$, $step_6$ i $step_7$ se dovlače bajtovi bloka iz memorije ***MEM*** u keš memoriju ***KEŠ***.

U slučaju operacije selektivnog vraćanja signal **CSC** je aktivan. Korak $step_1$ služi da se odgovarajući ulaz keš memorije ***KEŠ*** proglasi za nevažeći. To se realizuje generisanjem aktivne vrednosti signala **clearV** bloka *indikatori* ukoliko je aktivna vrednost signala **H/M** i neaktivna vrednost signala **D**. Iz koraka $step_1$ se prelazi ili na korak $step_2$ ili na korak $step_3$. U korak $step_3$ se prelazi ili ukoliko je signal **H/M** aktivan a signal **D** neaktivan ili ukoliko je signal **H/M** neaktivan. U korak $step_2$ se prelazi radi kompletiranja operacije. U korak $step_3$ se prelazi ukoliko su signali **H/M** i **D** aktivni. U koracima $step_3$ i $step_4$ se vraćaju bajtovi bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***, bit D datog bloka postavlja na neaktivnu vrednost i ponovo vraća na korak $step_1$.

U slučaju operacije kompletnog vraćanja signal **CCC** je aktivan. Korak $step_1$ služi da se odgovarajući ulaz keš memorije ***KEŠ*** proglasi za nevažeći i da se pređe na sledeći ulaz. To se postiže generisanjem aktivnih vrednosti signala **clearV** bloka *indikatori* i **incCNTCE** bloka *brojači* ukoliko je signal **D** neaktivan. Iz koraka $step_1$ se ili prelazi na korak $step_2$ ili na korak $step_3$ ili se ostaje u koraku $step_1$. U korak $step_3$ se prelazi ukoliko je signal **D** aktivan. U koracima $step_3$ i $step_4$ se vraćaju bajtovi bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***, bit **D** datog bloka postavlja na neaktivnu vrednost i ponovo vraća na korak $step_1$. U koraku $step_1$ se ostaje ukoliko su signali **D** i **CLEAN** neaktivni. U korak $step_2$ se prelazi ukoliko je signal **D** neaktivan, a signal **CLEAN** aktivan. U korak $step_2$ se prelazi radi kompletiranja operacije.

$step_2$: **CRP**,
if (**CRD**, **clCRD**),
if (**CWR**, **clCWR, writeDATA**, **writeD**),
if (**CSC**, **clCSC**),
if (**CCC**, **clCCC**),
br $step_0$

! U korak $step_2$ može da se dođe jedino iz koraka $step_1$ i to za sve četiri operacije. U koraku $step_2$ se bezuslovno generiše aktivna vrednost signala **CRP** koji keš memorija ***KEŠ*** preko bloka *cpu interfejs* šalje procesoru ***CPU***. Ukoliko je aktivna vrednost signala **CRD** generiše se aktivna vrednost signala **clCRD** bloka *cpu interfejs*, pa se na signal takta upisuje neaktivna vrednost u flip-flop CRD. Ukoliko je aktivna vrednost signala **CWR** generiše se aktivna vrednost signala **writeDATA** bloka *data memorija* kojim se upisuje nova vrednost u DATA memoriju. Pri tome se još generišu aktivne vrednosti signala **writeD** bloka *indikatori* i **clCWR** bloka *cpu interfejs*, pa se na signal takta upisuje aktivna vrednost u adresirani flip-flop D i neaktivna vrednost u flip-flop **CWR**. Ukoliko je aktivna vrednost signala **CSC,** generiše se aktivna vrednost signala **clCSC** bloka *cpu interfejs*, pa se na signal takta upisuje neaktivna vrednost u flip-flop CSC. Ukoliko je aktivna vrednost signala **CCC,** generiše se aktivna vrednost signala **clCCC** bloka *cpu interfejs*, pa se na signal takta upisuje neaktivna vrednost u flip-flop CCC. Na signal takta se uvek prelazi iz koraka $step_2$ u korak $step_0$.

$step_3$: **mxADATA**, **ldMDRWR**, **mxMAR**, **ldMAR**,
br $step_4$

! U korak $step_3$ se dolazi ili iz koraka $step_1$ ili iz koraka $step_4$. Iz koraka $step_1$ se dolazi ukoliko se u slučaju operacije čitanja ili upisa u njemu otkrije da nema saglasnosti i da je adresirani blok keš memorije ***KEŠ*** modifikovan pa ga treba vratiti iz keš memorije ***KEŠ*** u memoriju ***MEM***. Iz koraka $step_1$ se dolazi u slučaju operacije selektivnog vraćanja ukoliko se u njemu otkrije da postoji saglasnost i da je adresirani blok keš memorije ***KEŠ*** modifikovan pa ga treba vratiti iz keš memorije ***KEŠ*** u memoriju ***MEM***. Iz koraka $step_1$ se dolazi u slučaju operacije kompletnog vraćanja ukoliko se otkrije da je adresirani blok keš memorije ***KEŠ*** modifikovan pa ga treba vratiti iz keš memorije ***KEŠ*** u memoriju ***MEM***. Iz koraka $step_4$ se dolazi po prenosu jednog bajta bloka iz keš memorije ***KEŠ*** u memoriju ***MEM***.

U koraku $step_3$ se bezuslovno generišu aktivne vrednosti signala **mxADATA** bloka *data memorija* i signala **ldMDRWR**, **mxMAR** i **ldMAR** bloka *mem interfejs*. Aktivnom vrednošću signala **mxADATA** se u bloku *data memorija* kroz multiplekser propušta vrednost brojača CNTBB, dok se aktivnom vrednošću signala **ldMDRWR** obezbeđuje da se na signal takta u bloku *mem interfejs* podatak $DATARD_{7...0}$ očitan iz memorijskog modula DATA bloka *data memorija* upiše u registar MDRWR. U bloku *mem interfejs* se aktivnom vrednošću signala **mxMAR** kroz multiplekser propušta vrednost $TAGOUT_{10...0}$ očitana iz memorijskog modula TAG bloka *tag memorija*, dok se aktivnom vrednošću signala **ldMAR** na signal takta u registar MAR upisuje formirana adresa.

Na signal takta se uvek prelazi iz koraka $step_3$ na korak $step_4$.

$step_4$: **writeMEM**, **incMEMACC**,
if (**MEMFC**, **incCNTBB**),
if ((**MEMFC** · **CNTBB3** · (**CSC** + **CCC**), **clearD**),
br (*if* $\overline{\textbf{MEMFC}}$

then $step_4$
else (*if* $\overline{\textbf{CNTBB3}}$
then $step_3$
else (*if* (**CSC** + **CCC**) *then* $step_1$ *else* $step_5$)))

! U korak $step_4$ se dolazi samo iz koraka $step_3$ za sve četiri operacije ukoliko treba vratiti blok podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***.

U koraku $step_4$ se bezuslovno generišu aktivne vrednosti signala **writeMEM** bloka *mem interfejs* i **incMEMACC** bloka *brojači*. Aktivnom vrednošću signala **writeMEM** se sadržaj registra MDRWR bloka *mem interfejs* upisuje u memoriju ***MEM*** na adresi određenoj sadržajem registra MAR bloka *mem interfejs*. Aktivnom vrednošću signala **incMEMACC** se obezbeđuje da se pri pojavi signala takta inkrementira sadržaj brojača MEMACC koji određuje vreme pristupa memoriji ***MEM***. Aktivnom vrednošću signala **MEMFC** bloka *brojači* pri pojavi signala takta inkrementira se brojač CNTBB bloka *brojači*.

U slučaju operacija selektivnog vraćanja i kompletnog vraćanja aktivni su signali **CSC** i **CCC** bloka *cpu interfejs*, respektivno. Ukoliko su tada aktivne vrednosti signala **MEMFC** i **CNTBB3** bloka *brojači* generiše se aktivna vrednost signala **ClearD** bloka *indikatori*, kojim se pri pojavi signala takta bit D bloka *indikatori* adresiranog ulaza keš memorije ***KEŠ*** postavlja na neaktivnu vrednost. U slučaju operacija čitanja i upisa bit D adresiranog ulaza keš memorije ***KEŠ*** se postavlja na neaktivnu vrednost tek u koraku $step_7$ po dovlačenju novog bloka iz memorije ***MEM*** u dati ulaz keš memorije ***KEŠ***.

Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku $step_4$.

Pri aktivnoj vrednosti signala **MEMFC** na signal takta se prelazi iz koraka $step_4$ u korak $step_3$, $step_5$ ili $step_1$. U korak $step_3$ se prelazi za bilo koju od četiri operacije ukoliko nisu preneta iz keš memorije ***KEŠ*** u memoriju ***MEM*** sva četiri bajta bloka, pa brojač CNTBB ima sadržaj različit od tri. U korake $step_1$ i $step_5$ se prelazi ukoliko su preneti iz keš memorije ***KEŠ*** u memoriju ***MEM*** svi bajtovi bloka, pa brojač CNTBB ima sadržaj tri. U korak $step_1$ se prelazi u slučaju operacija selektivnog vraćanja i kompletnog vraćanja. Pošto je u koraku $step_3$ bit D adresiranog ulaza keš memorije ***KEŠ*** postavljen na neaktivnu vrednost, u koraku $step_1$ generisanjem aktivne vrednosti signala **clearV** bloka *indikatori* blok datog ulaza će biti proglašen za nevažeći. U korak $step_5$ se prelazi u slučaju operacija čitanja i upisa. U koracima $step_5$, $step_6$ i $step_7$ dovući će se bajtovi bloka iz memorije ***MEM*** u keš memoriju ***KEŠ*** i preći na korak $step_1$ u kome će sada biti otkrivena saglasnost.

$step_5$: **ldMAR**,
br $step_6$

! U korak $step_5$ se dolazi ili iz koraka step1 ili iz koraka $step_4$. Iz koraka $step_1$ se dolazi kada se otkrije nesaglasnost ali i utvrdi da sadržaj bloka nije modifikovan, pa se odmah prelazi na dovlačenje bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ***. Iz koraka $step_4$ se dolazi u slučaju operacija čitanja i upisa po završenom prenosu bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***. U koraku $step_5$ se bezuslovno generiše aktivna vrednost signala **ldMAR** bloka *mem interfejs*, čime se obezbeđuje da se na signal takta u registar MAR upiše formirana adresa. Na signal takta se uvek prelazi iz koraka $step_5$u korak $step_6$.

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

! U korak $step_6$ se dolazi samo iz koraka $step_5$. U koraku $step_6$ se bezuslovno generiše aktivna vrednost signala **incMEMACC** bloka *brojači* čime se obezbeđuje da se na signal takta inkrementira sadržaj brojača **MEMACC**, koji određuje vreme pristupa memoriji ***MEM***.

Pri neaktivnoj vrednosti signala **MEMFC** bloka *brojači* se ostaje u koraku $step_6$.

Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDRRD**, bloka *mem interfejs* čime se obezbeđuje da se na signal takta u registar MDRRD upiše vrednost očitana iz memorije ***MEM*** sa adrese određene sadržajem registra MAR bloka *mem interfejs*. Pri aktivnoj vrednosti signala **MEMFC** se na signal takta prelazi iz koraka $step_6$ u korak $step_7$.

$step_7$: **mxADATA**, **mxDIDATA**, **writeDATA**, **incCNTBB**,
if (**CNTBB3**, **writeTAG**, **writeV**, **clearD**),
br (*if* $\overline{\textbf{CNTBB3}}$ *then* $step_5$ *else* $step_1$)

! U korak $step_7$ se dolazi samo iz koraka $step_6$. U koraku $step_7$ se bezuslovno generišu aktivne vrednosti signala **mxADATA**, **mxDIDATA** i **writeDATA** bloka *data memorija* i signala **incCNTBB** bloka *brojači*. Aktivnim vrednostima signala **mxADATA** i **mxDIDATA** se kroz odgovarajuće multipleksere memorijskog modula DATA propuštaju sadržaj brojača $CNTBB_{1...0}$ i sadržaj registra $MDRRD_{7...0}$, respektivno. Aktivnom vrednošću signala **writeDATA** se vrši upis u memorijski modul DATA. Aktivnom vrednošću signala

incCNTBB se na signal takta inkrementira sadržaj brojača CNTBB. U koraku $step_7$ se generišu i aktivne vrednosti signala **writeTAG** bloka *tag memorija* i signala **writeV** i **clearD** bloka *brojači* ukoliko je sadržaj brojača CNTBB jednak tri. Aktivnom vrednošću signala **writeTAG** se vrši upis broja grupe generisane adrese u adresiranu lokaciju memorijskog modula TAG. Aktivnom vrednošću signala **writeV** se na signal takta adresirani flip-flop V postavlja na aktivnu vrednost. Aktivnom vrednošću signala **clearD** se na signal takta adresirani flip-flop D postavlja na neaktivnu vrednost. Na signal takta se prelazi iz koraka $step_7$ u korak $step_5$ ukoliko brojač CNTBB ima sadržaj različit od tri, odnosno u korak $step_1$ ako je njegov sadržaj tri.

1.2.1.3.2.3. Vremenski oblici signala

U keš memoriji ***KEŠ*** mogu da se izvršavaju operacije čitanja, upisa, selektivnog vraćanja i kompletnog vraćanja. Izvršavanje svake operacije se sastoji od nekoliko aktivnosti čija realizacija zahteva jedan ili više koraka. Te aktivnosti i koraci u kojima se one realizuju su:

1. *Čekanje signala startovanja operacije čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja*. Ova aktivnost se realizuje u koraku $step_0$.
2. *Provera da li ima saglasnosti*. Ova aktivnost se realizuje u koraku $step_1$.
3. *Provera da li je blok modifikovan*. Ova aktivnost se realizuje u koraku $step_1$.
4. *Čitanje bajta podatka*. Ova aktivnost se realizuje u koraku $step_1$.
5. *Slanje indikacije o završenoj operaciji čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja*. Ova aktivnost se realizuje u koraku $step_2$.
6. *Upis bajta podatka*. Ova aktivnost se realizuje u koraku $step_2$.
7. *Vraćanje bajtova bloka*. Ova aktivnost se realizuje u koracima $step_3$ i $step_4$.
8. *Dovlačenje bajtova bloka*. Ova aktivnost se realizuje u koracima $step_5$, $step_6$ i $step_7$.

Vremenski oblici signala za operacije čitanja, upisa, selektivnog vraćanja i kompletnog vraćanja za različite situacije u kojima keš memorija ***KEŠ*** može da se nađe dobijaju se kombinovanjem vremenskih oblika signala za osam navedenih aktivnosti.

Vremenski oblici signala za svaku od ove četiri operacije i to za različite situacije u kojima pri tome keš memorija ***KEŠ*** može da se nađe dati su u daljem tekstu.

1.2.1.3.2.3.1. Operacija čitanja

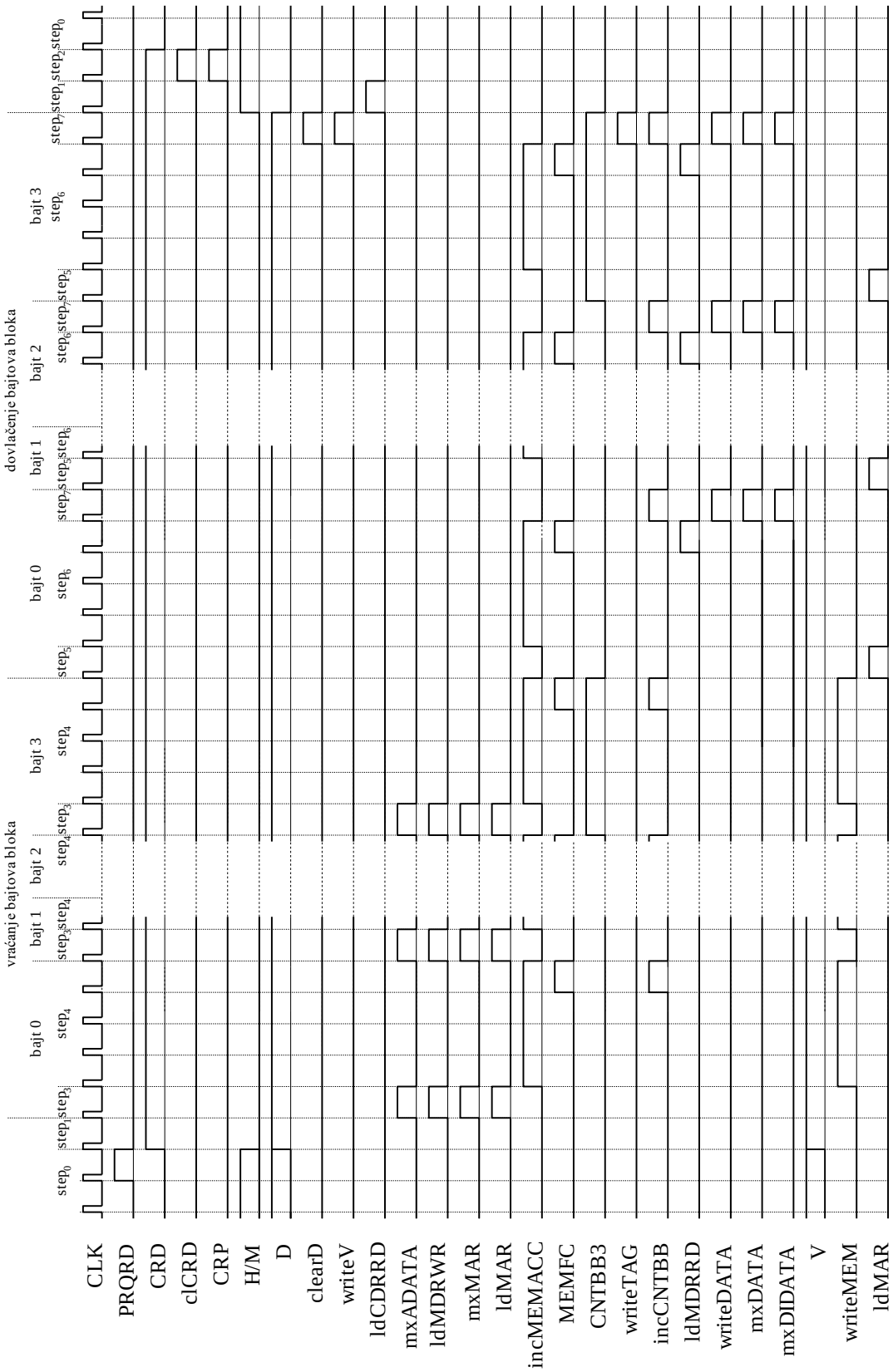
U slučaju operacije čitanja mogu da se jave četiri slučaja:

- nema saglasnosti, ulaz je važeći i blok je modifikovan,
- nema saglasnosti, ulaz je važeći i blok nije modifikovan,
- nema saglasnosti i ulaz nije važeći i
- ima saglasnosti.

Vremenski oblici signala za sva četiri slučaja dati su u daljem tekstu.

Vremenski oblici signala za slučaj kada se u keš memoriji ***KEŠ*** izvršava operacija čitanja, nema saglasnosti, ulaz je važeći i blok je modifikovan dati su na slici 25.

Keš memorija ***KEŠ*** u koraku $step_0$ čeka pojavu aktivne vrednosti signala startovanja operacije čitanja **PRQRD** koji preko bloka *cpu interfejs* dolazi iz procesora ***CPU***. Pri njenoj pojavi na signal takta se u registar CAR bloka *cpu interfejs* upisuje adresa memorije ***MEM*** sa koje treba očitati bajt podatka, flip-flop CRD bloka *cpu interfejs* se postavlja na aktivnu vrednost i prelazi se u korak $step_1$.



Slika 25 Vremenski oblici signala za slučaj operacija čitanja, nema saglasnosti, ulaz važeći i blok modifikovan

Do pojave signala takta kojim se u registar CAR upisuje nova adresa u registru CAR je bila vrednost od prethodne operacije čitanja, upisa ili selektivnog vraćanja. Zbog toga su u koraku

$step_0$ nedefinisane vrednosti signala **V** bloka *indikatori*, **H/M** bloka *tag memorija* i **D** bloka *indikatori*. Prelaskom na korak $step_1$ ovi signali dobijaju definisane vrednosti. Uzeto je da je signal **V** aktivan, što znači da ju ulaz važeći, da je signal **H/M** neaktivan, što znači da se u datom ulazu keš memorije ***KEŠ*** nalazi blok iz neke druge grupe memorije ***MEM***, i da je signal **D** aktivan, što znači da je blok iz datog ulaza keš memorije ***KEŠ*** modifikovan. Zbog toga je potrebno, najpre, četiri puta proći kroz korake $step_3$ i $step_4$ radi vraćanja četiri bajta bloka iz keš memorije ***KEŠ*** u memoriju ***MEM***, zatim četiri puta proći kroz korake $step_5$, $step_6$ i $step_7$ radi dovlačenja četiri bajta bloka iz memorije ***MEM*** u keš memoriju ***KEŠ*** i na kraju ponovo u koraku $step_1$ formirati vrednosti signala **V**, **H/M** i **D**.

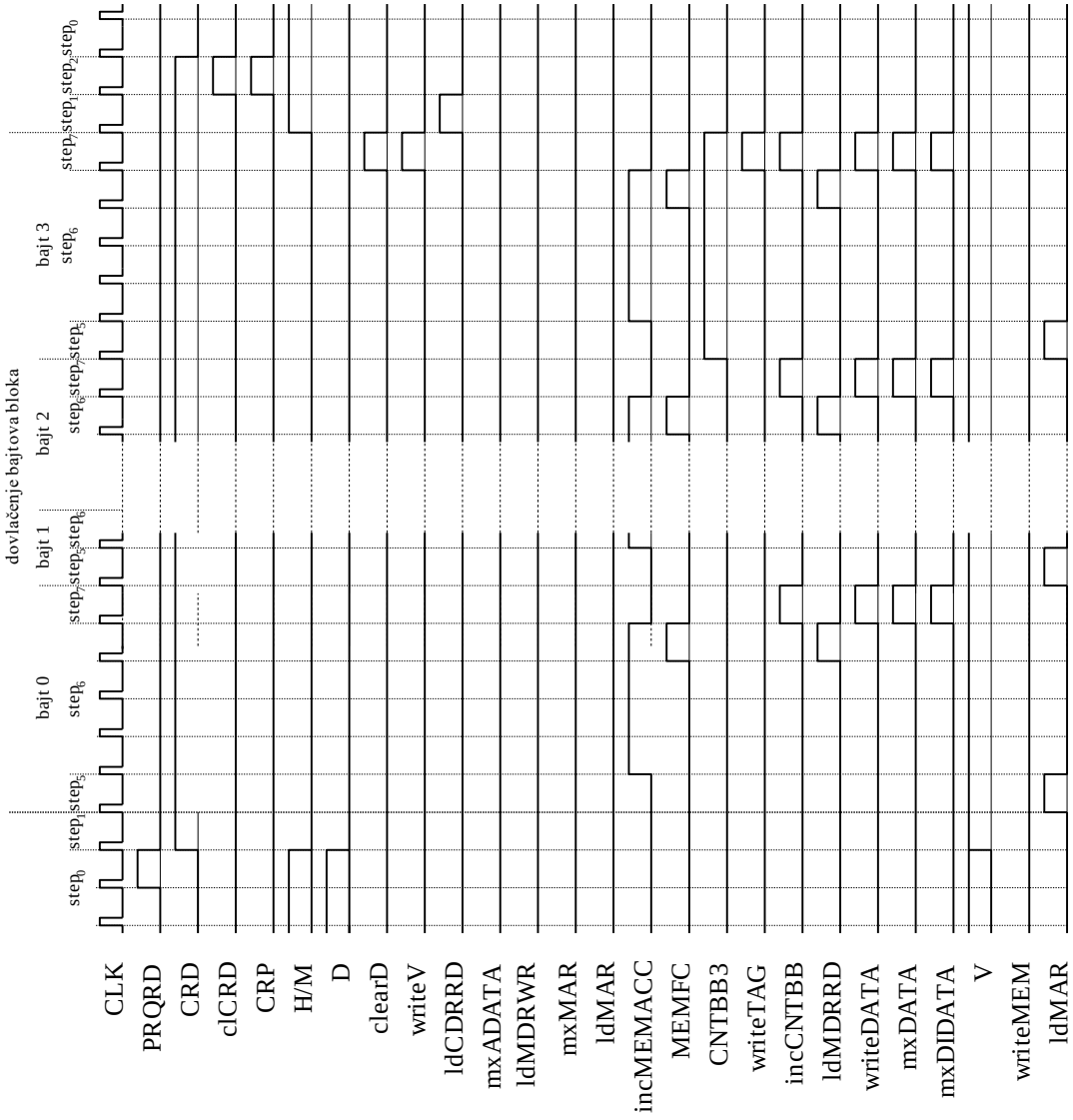
U koraku $step_3$ se generišu aktivne vrednosti signala **mxADATA** bloka *data memorija* i **ldMDRWR**, **mxMAR** i **ldMAR** bloka *mem interfejs* i time u registar MDRWR i MAR upisuju podatak i adresa za upis u memoriju ***MEM***. U koraku $step_4$ se generiše aktivna vrednost signala **writeMEM** bloka *mem interfejs*, čime se realizuje upis u memoriju ***MEM*** podatka iz registra MDRWR na adresi određenoj sadržajem registra MAR. U koraku $step_4$ je aktivna vrednost i signala **incMEMACC** bloka *brojači*, čime se omogućava inkrementiranje sadržaja brojača MEMACC. Kada sadržaj brojača MEMACC postane tri, generišu se aktivne vrednosti signala **MEMFC** i **incCNTBB** bloka *brojači*. Aktivna vrednost signala **MEMFC** omogućuje prelazak iz koraka $step_4$ ili na korak $step_3$ ili na korak $step_5$. Aktivna vrednost signala **incCNTBB** omogućuje inkrementiranja brojača CNTBB. Sve dok je signal **CNTBB3** bloka *brojači* neaktivan prelazi se iz koraka $step_4$ na korak $step_3$, a kada signal **CNTBB3** postane aktivan prelazi se iz koraka $step_4$ na korak $step_5$. Signal **CNTBB3** je aktivan kada se po četvri put prolazi kroz korake $step_3$ i $step_4$. Prelaskom iz koraka $step_4$ na korak $step_5$, završava se prebacivanje četiri bajta bloka iz keš memorije ***KEŠ*** u memoriju ***MEM*** i započinje dovlačenje četiri bajta bloka iz memorije ***MEM*** u keš memoriju ***KEŠ***.

U koraku $step_5$ se generiše aktivna vrednost signala **ldMAR** i time u registar MAR upisuju adresa za čitanje iz memorije ***MEM***. Iz koraka $step_5$ se prelazi na korak $step_6$. U koraku $step_6$ se realizuje čitanje podatka iz memorije ***MEM*** sa adrese određene sadržajem registra MAR i upis u registar MDRRD bloka *mem interfejs*. U koraku $step_6$ je aktivna vrednost signala **incMEMACC**, čime se omogućava inkrementiranje sadržaja brojača MEMACC. Kada sadržaj brojača MEMACC postane tri, generišu se aktivne vrednosti signala **ldMDRRD** bloka *mem interfejs* i **MEMFC** bloka *brojači*. Aktivna vrednost signala **ldMDRRD** omogućuje upis u registar MDRRD podatka očitanog iz memorije ***MEM*** sa adrese određene sadržajem registra MAR. Aktivna vrednost signala **MEMFC** omogućuje prelazak iz koraka $step_6$na korak $step_7$. U koraku $step_7$ se generišu aktivne vrednosti signala **mxADATA**, **mxDIDATA** i **writeDATA** bloka *data memorija* i **incCNTBB** bloka *brojači*. Aktivne vrednosti signala **mxADATA**, **mxDIDATA** i **write DATA** omogućuju upis očitanog bajta podatka iz registra MDRRD u memorijski modul DATA. Aktivna vrednost signala **incCNTBB** omogućuje inkrementiranja brojača CNTBB. Sve dok je signal **CNTBB3** neaktivan prelazi se iz koraka $step_7$ na korak $step_5$, a kada signal **CNTBB3** postane aktivan prelazi se iz koraka $step_7$ na korak $step_1$. Signal **CNTBB3** je aktivan kada se po četvri put prolazi kroz korake $step_5$, $step_6$ i $step_7$. Tada se u koraku $step_7$ generišu i aktivne vrednosti signala **writeTAG** bloka *tag memorija* i **writeV** i **clearD** bloka *indikatori*. Signalom **writeTAG** se u memorijski modul TAG upisuje broj grupe bloka memorije ***MEM*** koji je dovučen u dati ulaz keš memorije ***KEŠ***, signalom **writeV** se indikator V datog ulaza keš memorije ***KEŠ*** postavlja na aktivnu vrednost i signalom **clearD** se indikator D datog ulaza keš memorije ***KEŠ*** postavlja na neaktivnu vrednost. Prelaskom iz koraka $step_7$ na korak $step_1$, završava se prebacivanje četiri bajta bloka iz memorije ***MEM*** u keš memoriju ***KEŠ***.

U koraku $step_1$ se ponovo vrši provera da li ima saglasnosti i dobija da je signal **V** aktivan, što znači da je ulaz važeći, da je signal **H/M** aktivan, što znači da je u datom ulazu keš

memorije ***KEŠ*** otkrivena saglasnost, i da je signal **D** neaktivan, što znači da blok iz datog ulaza keš memorije ***KEŠ*** nije modifikovan. Pored toga u koraku $step_1$ se generiše i aktivna vrednost signala **ldCDRRD** čime se u registar CDRRD upisuje bajt podatka očitan iz keš memorije **KEŠ**. Iz koraka $step_1$ se prelazi na korak $step_2$. U koraku $step_2$ se generišu aktivne vrednosti signala **CRP** i **clCRD** bloka *cpu interfejs*. Signalom **CRP** se procesoru ***CPU*** šalje indikacija da je operacija čitanja završena, a signalom **clCRD** se flip-flop CRD postavlja na neaktivnu vrednost. Iz koraka $step_2$ se prelazi na korak $step_0$. U koraku $step_0$ se čeka pojava aktivne vrednosti signala startovanja operacije čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja.

Vremenski oblici signala za slučaj kada se u keš memoriji izvršava operacija čitanja, nema saglasnosti, ulaz je važeći i blok nije modifikovan dati su na slici 26.



Slika 26 Vremenski oblici signala za slučaj operacija čitanja, nema saglasnosti i blok nije modifikovan

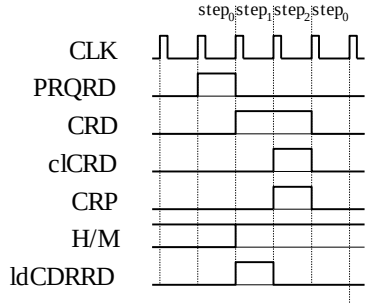
Vremenski oblici signala su veoma slični sa onima za prethodno opisani slučaj. Uzeto je da se u koraku $step_1$ utvrđuje da je signal **V** aktivan, što znači da ju ulaz važeći, da je signal **H/M** neaktivan, što znači da se u datom ulazu keš memorije ***KEŠ*** nalazi blok iz neke druge grupe memorije ***MEM***, i da je signal **D** neaktivan, što znači da blok iz datog ulaza keš memorije

KEŠ nije modifikovan. Zbog toga se iz koraka $step_1$ prelazi na korak $step_5$, potom četiri puta prolazi kroz korake $step_5$, $step_6$ i $step_7$ radi dovlačenja četiri bajta bloka iz memorije ***MEM*** u keš memoriju ***KEŠ*** i na kraju prelazi ponovo u korak $step_1$ radi formiranja vrednosti signala **V**, **H/M** i **D**. Pošto je sada signal **H/M** aktivan, što znači da je otkrivena saglasnost, aktivnosti su iste i generišu se isti signali kao i u prethodno opisanom slučaju. To se realizuje prelaskom iz koraka $step_1$ na korak $step_2$ i na kraju u korak $step_0$.

Vremenski oblici signala za slučaj kada se u keš memoriji izvršava operacija čitanja, nema saglasnosti i ulaz nije važeći su veoma veoma slični vremenskim oblicima signala za slučaj kada se u keš memoriji izvršava operacija čitanja, nema saglasnosti, ulaz je važeći i blok nije modifikovan (slika 26) pa zato nisu dati posebno. U ovom slučaju se u koraku $step_1$ utvrđuje da je pored signala **H/M** i **D** i signal **V** neaktivan. Signal **V** ostaje neaktivan sve dok se u koracima $step_5$, $step_6$ i $step_7$ dovlače četiri bajta bloka iz memorije ***MEM*** u keš memoriju ***KEŠ***, a postaje aktivan pri ponovom prelasku u korak $step_1$ radi ponovnog formiranja vrednosti signala **V**, **H/M** i **D**.

Vremenski oblici signala za slučaj kada se u keš memoriji izvršava operacija čitanja i pri tome ima saglasnosti dati su na slici 27.

Keš memorija ***KEŠ*** u koraku $step_0$ čeka pojavu aktivne vrednosti signala startovanja operacije čitanja **PRQRD** koji preko bloka *cpu interfejs* dolazi iz procesora ***CPU***. Pri njenoj pojavi na signal takta se u registar CAR bloka *cpu interfejs* upisuje adresa memorije ***MEM*** sa koje treba očitati bajt podatka, flip-flop CRD bloka *cpu interfejs* se postavlja na aktivnu vrednost i prelazi se u korak $step_1$. U koraku $step_1$ se dobija da je signal **H/M** bloka *tag memorija* aktivan, što znači da je u datom ulazu keš memorije ***KEŠ*** otkrivena saglasnost. Zbog toga se u koraku $step_1$ se generiše aktivna vrednost signala **ldCDRRD** bloka *cpu interfejs* čime se u registar CDRRD upisuje bajt podatka očitan iz keš memorije ***KEŠ***. Iz koraka $step_1$ se prelazi na korak $step_2$. U koraku $step_2$ se generišu aktivne vrednosti signala **CRP** i **clCRD** bloka *cpu interfejs*. Signalom **CRP** se procesoru ***CPU*** šalje indikacija da je operacija čitanja završena, a signalom **clCRD** se flip-flop CRD postavlja na neaktivnu vrednost. Iz koraka $step_2$ se prelazi na korak $step_0$. U koraku $step_0$ se čeka pojava aktivne vrednosti signala startovanja operacije čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja.



Slika 27 Vremenski oblici signala za slučaj operacija čitanja i ima saglasnosti

1.2.1.3.2.3.2. Operacija upisa

U slučaju operacije upisa mogu da se jave četiri slučaja:

- nema saglasnosti, ulaz je važeći i blok je modifikovan,
- nema saglasnosti, ulaz je važeći i blok nije modifikovan,
- nema saglasnosti i ulaz nije važeći i
- ima saglasnosti.

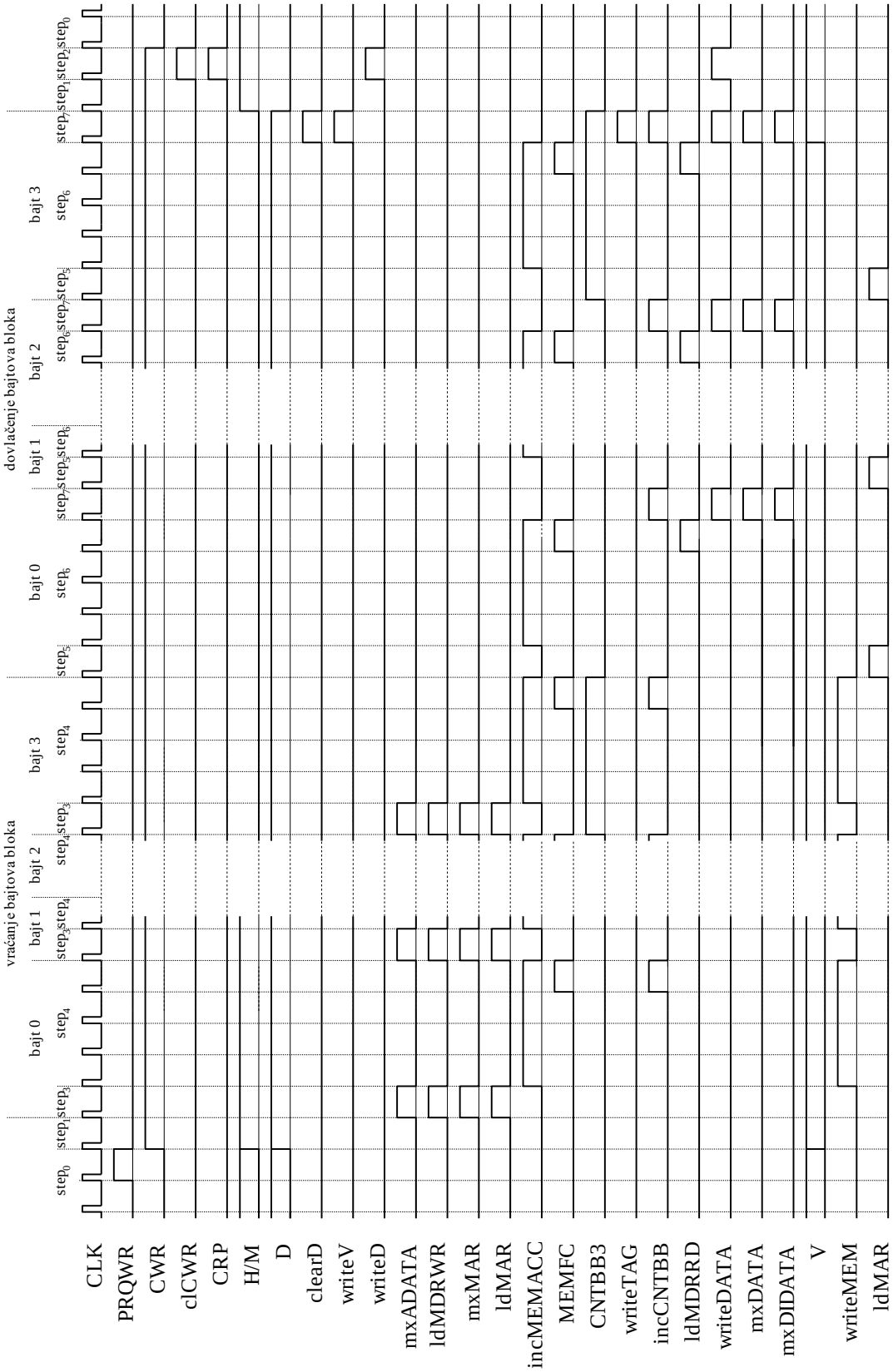
Vremenski oblici signala za sva četiri slučaja dati su u daljem tekstu.

Vremenski oblici signala za slučaj kada se u keš memoriji izvršava operacija upisa, nema saglasnosti, ulaz je važeći i blok je modifikovan dati su na slici 28 .

Keš memorija ***KEŠ*** u koraku $step_0$ čeka pojavu aktivne vrednosti signala startovanja operacije čitanja **PRQWR** koji preko bloka *cpu interfejs* dolazi iz procesora ***CPU***. Pri njenoj pojavi na signal takta se u registre CAR i CDRWR bloka *cpu interfejs* upisuju adresa memorije ***MEM*** na kojoj treba upisati bajt podatka i sam bajt podatka, respektivno, flip-flop CWR se postavlja na aktivnu vrednost i prelazi se u korak $step_1$.

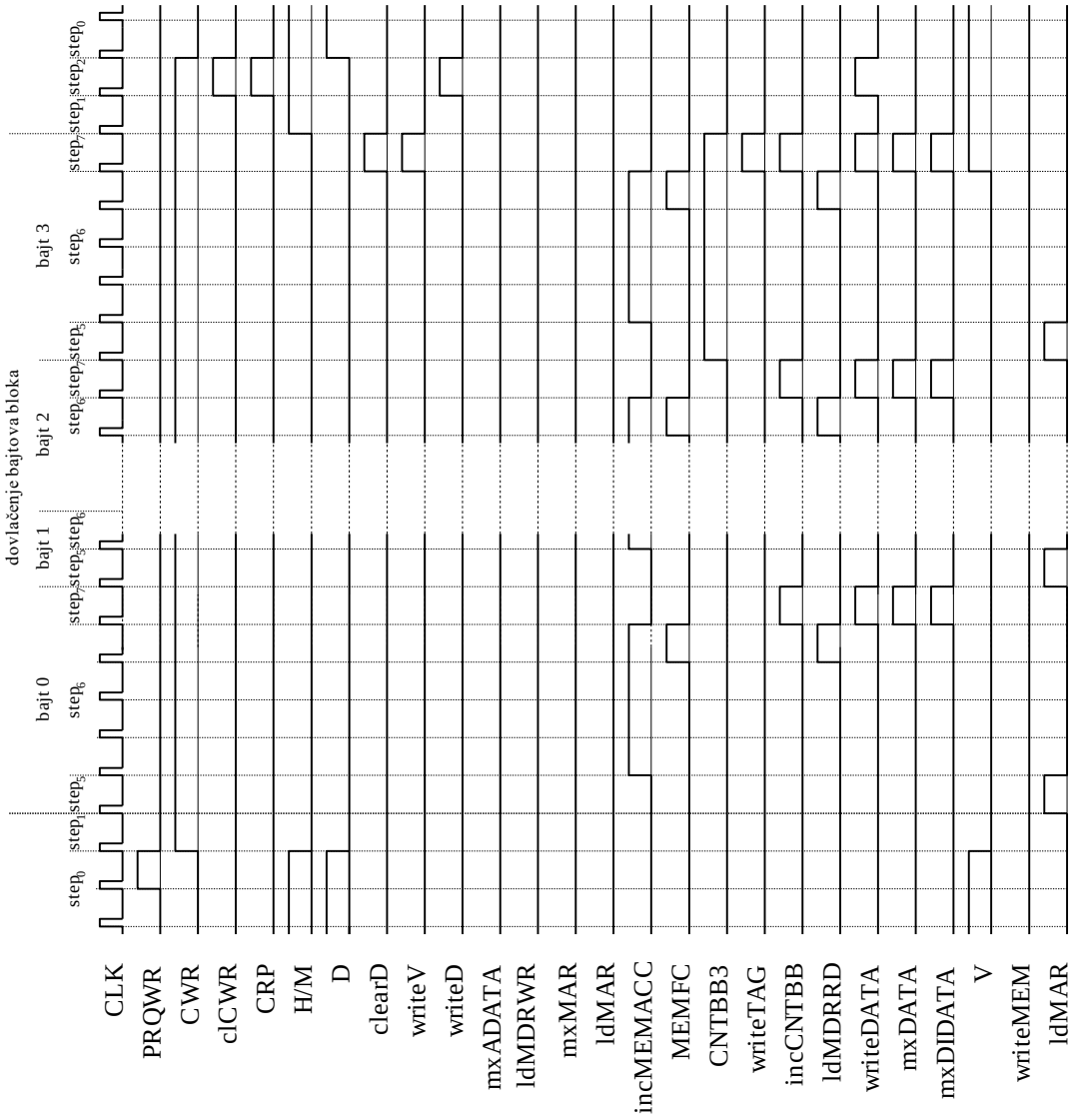
Do pojave signala takta kojim se u registar CAR upisuje nova adresa u registru CAR je bila vrednost od prethodne operacije čitanja, upisa ili selektivnog vraćanja. Zbog toga su u koraku $step_0$ nedefinisane vrednosti signala **V** bloka *indikatori*, **H/M** bloka *tag memorija* i **D** bloka *indikatori*. Prelaskom na korak $step_1$ ovi signali dobijaju definisane vrednosti. Uzeto je da je signal **V** aktivan, što znači da ju ulaz važeći, da je signal **H/M** neaktivan, što znači da se u datom ulazu keš memorije ***KEŠ*** nalazi blok iz neke druge grupe memorije ***MEM***, i da je signal **D** aktivan, što znači da je blok iz datog ulaza keš memorije ***KEŠ*** modifikovan. Zbog toga je potrebno, najpre, četiri puta proći kroz korake $step_3$ i $step_4$ radi vraćanja četiri bajta bloka iz keš memorije ***KEŠ*** u memoriju ***MEM***, zatim četiri puta proći kroz korake $step_5$, $step_6$ i $step_7$ radi dovlačenja četiri bajta bloka iz memorije ***MEM*** u keš memoriju ***KEŠ*** i na kraju ponovo u koraku $step_1$ formirati vrednosti signala **V**, **H/M** i **D**. Vraćanja četiri bajta bloka iz keš memorije ***KEŠ*** u memoriju ***MEM*** i dovlačenja četiri bajta bloka iz memorije ***MEM*** u keš memoriju ***KEŠ*** se realizuje na identičan način kao i u slučaju operacija čitanja, nema saglasnosti, ulaz je važeći i blok je modifikovan. Zbog toga su sva objašnjenja vezana za aktivnosti i generisanje signala u koracima $step_3$ i $step_4$ i koracima $step_5$, $step_6$ i $step_7$ identična sa odgovarajućim objašnjenjima iz odeljka 1.2.1.3.2.3.1. Po završetku prebacivanje četiri bajta bloka iz memorije ***MEM*** u keš memoriju ***KEŠ*** prelazi se iz koraka $step_7$ na korak $step_1$.

U koraku $step_1$ se ponovo vrši provera da li ima saglasnosti i dobija da je signal **V** aktivan, što znači da je ulaz važeći, da je signal **H/M** aktivan, što znači da je u datom ulazu keš memorije ***KEŠ*** otkrivena saglasnost, i da je signal **D** neaktivan, što znači da blok iz datog ulaza keš memorije ***KEŠ*** nije modifikovan. Iz koraka $step_1$ se prelazi na korak $step_2$. U koraku $step_2$ se generišu aktivne vrednosti signala **CRP** i **clCWR** bloka *cpu interfejs*, **writeDATA** bloka *data memorija* i **writeD** bloka *indikatori*. Signalom **CRP** se procesoru ***CPU*** šalje indikacija da je operacija čitanja završena, signalom **clCWR** se flip-flop CWR postavlja na neaktivnu vrednost, signalom **writeDATA** se bajt podatka upisuje u memorijski modul DATA i signalom **writeD** se indikator D odgovarajućeg ulaza keš memorije ***KEŠ*** postavlja na aktivnu vrednost. Iz koraka $step_2$ se prelazi na korak $step_0$. U koraku $step_0$ se čeka pojava aktivne vrednosti signala startovanja operacije čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja.



Slika 28 Vremenski oblici signala za slučaj operacija upisa, nema saglasnosti, ulaz je važeći i blok je modifikovan

Vremenski oblici signala za slučaj kada se u keš memoriji izvršava operacija upisa, nema saglasnosti, ulaz je važeći i blok nije modifikovan dati su na slici 29.



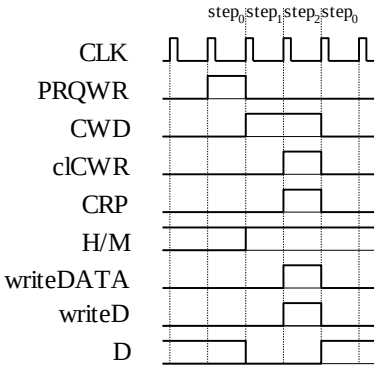
Slika 29 Vremenski oblici signala za slučaj operacija upisa, nema saglasnosti, ulaz je važeći i blok nije modifikovan

Vremenski oblici signala su veoma slični sa onima za prethodno opisani slučaj. Uzeto je da se u koraku $step_1$ utvrđuje da je signal **V** aktivan, što znači da je ulaz važeći, da je signal **H/M** neaktivan, što znači da se u datom ulazu keš memorije ***KEŠ*** nalazi blok iz neke druge grupe memorije ***MEM***, i da je signal **D** neaktivan, što znači da blok iz datog ulaza keš memorije ***KEŠ*** nije modifikovan. Zbog toga se iz koraka $step_1$ prelazi na korak $step_5$, potom četiri puta prolazi kroz korake $step_5$, $step_6$ i $step_7$ radi dovlačenja četiri bajta bloka iz memorije ***MEM*** u keš memoriju ***KEŠ*** i na kraju prelazi ponovo u korak $step_1$ radi formiranja vrednosti signala **V**, **H/M** i **D**. Pošto je sada signal **H/M** aktivan, što znači da je otkrivena saglasnost, aktivnosti su iste i generišu se isti signali kao i u prethodno opisanom slučaju. To se realizuje prelaskom iz koraka $step_1$ na korak $step_2$ i na kraju u korak $step_0$.

Vremenski oblici signala za slučaj kada se u keš memoriji izvršava operacija upisa, nema saglasnosti i ulaz nije važeći su veoma slični vremenskim oblicima signala za slučaj kada se u keš memoriji izvršava operacija upisa, nema saglasnosti, ulaz je važeći i blok nije modifikovan (slika 29) pa zato nisu dati posebno. U ovom slučaju se u koraku $step_1$ utvrđuje da je pored signala **H/M** i **D** i signal **V** neaktivan. Signal **V** ostaje neaktivan sve dok se u koracima $step_5$, $step_6$ i $step_7$ dovlače četiri bajta bloka iz memorije ***MEM*** u keš memoriju

KEŠ, a postaje aktivan pri ponovom prelasku u korak $step_1$ radi ponovnog formiranja vrednosti signala **V**, **H/M** i **D**.

Vremenski oblici signala za slučaj kada se u keš memoriji izvršava operacija upisa i pri tome ima saglasnosti dati su na slici 30.



Slika 30 Vremenski oblici signala za slučaj operacija upisa i ima saglasnosti

Keš memorija ***KEŠ*** u koraku $step_0$ čeka pojavu aktivne vrednosti signala startovanja operacije čitanja **PRQWR** koji preko bloka *cpu interfejs* dolazi iz procesora ***CPU***. Pri njenoj pojavi na signal takta se u registre CAR i CDRWR bloka *cpu interfejs* upisuju adresa memorije ***MEM*** na kojoj treba upisati bajt podatka i sam bajt podatka, respektivno, flip-flop CWR bloka *cpu interfejs* se postavlja na aktivnu vrednost i prelazi se u korak $step_1$. U koraku $step_1$ se dobija da je signal **H/M** bloka *tag memorija* aktivan, što znači da je u datom ulazu keš memorije ***KEŠ*** otkrivena saglasnost. Stoga se iz koraka $step_1$ prelazi na korak $step_2$. U koraku $step_2$ se generišu aktivne vrednosti signala **CRP** i **clCWR** bloka *cpu interfejs* i **writeDATA** i **writeD** bloka *data memorija*. Signalom **CRP** se procesoru ***CPU*** šalje indikacija da je operacija čitanja završena, signalom **clCWR** se flip-flop CWR postavlja na neaktivnu vrednost, signalom **writeDATA** se bajt podatka upisuje u memorijski modul DATA i signalom **writeD** se indikator D odgovarajućeg ulaza keš memorije ***KEŠ*** postavlja na aktivnu vrednost. Iz koraka $step_2$ se prelazi na korak $step_0$. U koraku $step_0$ se čeka pojava aktivne vrednosti signala startovanja operacije čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja.

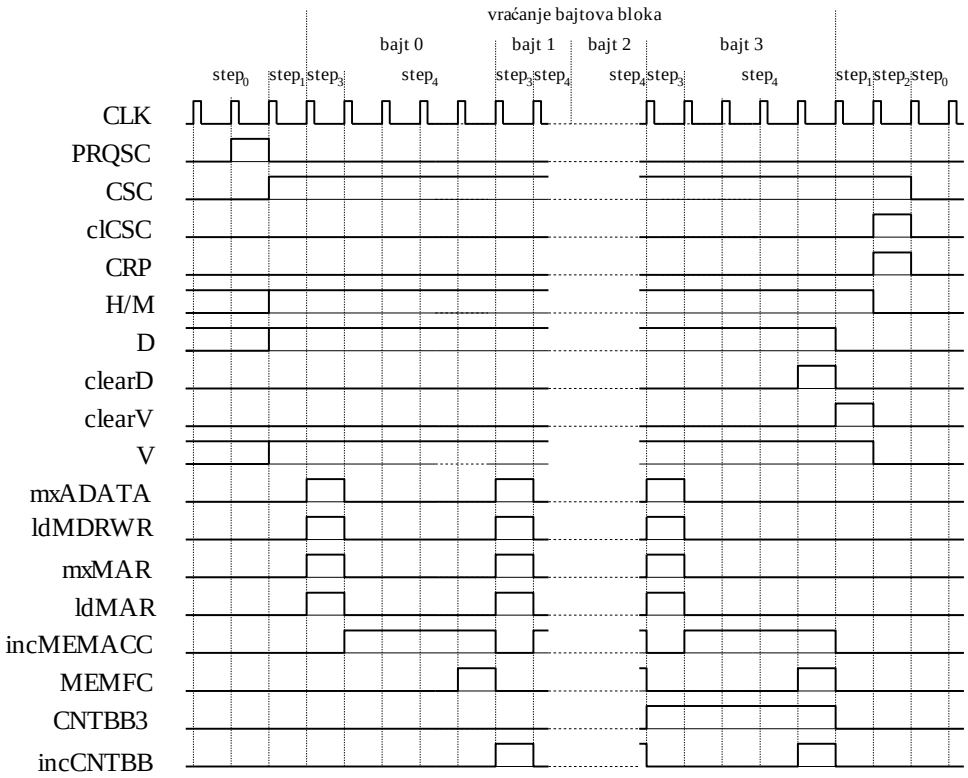
1.2.1.3.2.3.3. Operacija selektivnog vraćanja

U slučaju operacije selektivnog vraćanja mogu da se jave tri slučaja:

- ima saglasnosti i blok je modifikovan,
- ima saglasnosti i blok nije modifikovan i
- nema saglasnosti.

Vremenski oblici signala za sva tri slučaja daju su u daljem tekstu.

Vremenski oblici signala operacije selektivnog vraćanja za slučaj kada u keš memoriji ima saglasnosti i blok je modifikovan dati su na slici 31.

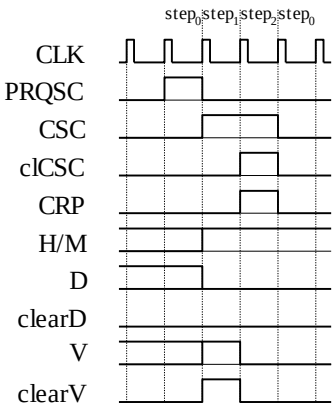


Slika 31 Vremenski oblici signala za slučaj operacija selektivno vraćanje, ima saglasnosti i blok modifikovan

Keš memorija ***KEŠ*** u koraku $step_0$ čeka pojavu aktivne vrednosti signala startovanja operacije selektivnog vraćanja **PRQSC** koji preko bloka *cpu interfejs* dolazi iz procesora ***CPU***. Pri njenoj pojavi na signal takta se u registar CAR bloka *cpu interfejs* upisuje adresa memorije ***MEM*** za koju treba proveriti da li se blok podataka kome pripada i podatak sa date adrese nalazi u keš memoriji ***KEŠ***, flip-flop CSC bloka *cpu interfejs* se postavlja na aktivnu vrednost i prelazi se u korak $step_1$. Do pojave signala takta kojim se u registar CAR upisuje nova adresa u registru CAR je bila vrednost od prethodne operacije čitanja, upisa ili selektivnog vraćanja. Zbog toga su u koraku $step_0$ nedefinisane vrednosti signala **V** bloka *indikatori*, **H/M** bloka *tag memorija* i **D** bloka *indikatori*. Prelaskom na korak $step_1$ ovi signali dobijaju definisane vrednosti. Uzeto je da je signal **V** aktivan, što znači da je ulaz važeći, da je signal **H/M** aktivan, što znači da se u datom ulazu keš memorije ***KEŠ*** nalazi blok iz grupe memorije ***MEM*** kojoj pripada i generisana adresa, i da je signal **D** aktivan, što znači da je blok iz datog ulaza keš memorije ***KEŠ*** modifikovan. Zbog toga je potrebno, najpre, četiri puta proći kroz korake $step_3$ i $step_4$ radi vraćanja četiri bajta bloka iz keš memorije ***KEŠ*** u memoriju ***MEM***, i na kraju ponovo u koraku $step_1$ formirati vrednosti signala **V**, **H/M** i **D**. Vraćanja četiri bajta bloka iz keš memorije ***KEŠ*** u memoriju ***MEM*** se realizuje na identičan način kao i u slučaju operacija čitanja, nema saglasnosti, ulaz je važeći i blok je modifikovan. Zbog toga su sva objašnjenja vezana za aktivnosti i generisanje signala u koracima $step_3$ i $step_4$ i identična sa odgovarajućim objašnjenjima iz odeljka 1.2.1.3.2.3.1. Po završetku prebacivanje četiri bajta bloka iz keš memorije ***KEŠ*** u memoriju ***MEM*** prelazi se iz koraka $step_4$ na korak $step_1$. U koraku $step_1$ se sada dobija da je signal **V** aktivan, što znači da je ulaz važeći, da je signal **H/M** aktivan, što znači da se u datom ulazu keš memorije ***KEŠ*** nalazi blok iz grupe memorije ***MEM*** kojoj pripada i generisana adresa, i da je signal **D** neaktivan, što znači da blok iz datog ulaza keš memorije ***KEŠ*** nije modifikovan. Ova kombinacija signala **V**, **H/M** i **D** u slučaju operacije selektivnog vraćanja označava i da se u datom ulazu keš memorije ***KEŠ*** nalazi blok iz grupe memorije ***MEM*** kojoj pripada i generisana adresa, da

između sadržaja ovog bloka u keš memoriji ***KEŠ*** i memoriji ***MEM*** ne postoji razlika i da sada dati ulaz keš memorije ***KEŠ*** treba proglasiti za nevažeći. Zbog toga se u koraku $step_1$ generiše aktivna vrednost signala **clearV** bloka *indikatori*, čime se u odgovarajući flip-flop V upisuje neaktivna vrednost, i potom prelazi iz koraka $step_1$ na korak $step_2$. U koraku $step_2$ se generišu aktivne vrednosti signala **CRP** i **clCSC** bloka *cpu interfejs*. Signalom **CRP** se procesoru ***CPU*** šalje indikacija da je operacija selektivnog vraćanja završena, a signalom **clCSC** se flip-flop CSC postavlja na neaktivnu vrednost. Iz koraka $step_2$ se prelazi na korak $step_0$. U koraku $step_0$ se čeka pojava aktivne vrednosti signala startovanja operacije čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja.

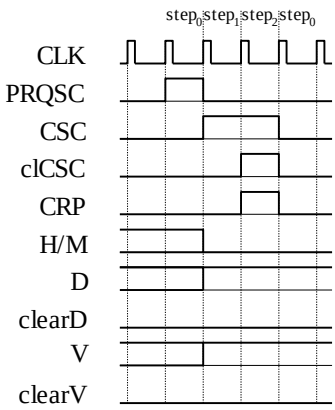
Vremenski oblici signala operacije selektivnog vraćanja za slučaj kada u keš memoriji ima saglasnosti i blok nije modifikovan dati su na slici 32.



Slika 32 Vremenski oblici signala za slučaj operacija selektivno vraćanje, ima saglasnosti i blok nije modifikovan

Keš memorija ***KEŠ*** u koraku $step_0$ čeka pojavu aktivne vrednosti signala startovanja operacije selektivnog vraćanja. Pri njenoj pojavi na signal takta se u registar CAR upisuje adresa memorije ***MEM*** za koju treba proveriti da li se blok podataka kome pripada i podatak sa date adrese nalazi u keš memoriji ***KEŠ***, flip-flop CSC se postavlja na aktivnu vrednost i prelazi se u korak $step_1$. U koraku $step_1$ se dobija da je signal **V** aktivan, što znači da je ulaz važeći, da je signal **H/M** aktivan, što znači da se u datom ulazu keš memorije ***KEŠ*** nalazi blok iz grupe memorije ***MEM*** kojoj pripada i generisana adresa, i da je signal **D** neaktivan, što znači da blok iz datog ulaza keš memorije ***KEŠ*** nije modifikovan. Ova kombinacija signala **V**, **H/M** i **D** u slučaju operacije selektivnog vraćanja označava i da se u datom ulazu keš memorije ***KEŠ*** nalazi blok iz grupe memorije ***MEM*** kojoj pripada i generisana adresa, da između sadržaja ovog bloka u keš memoriji ***KEŠ*** i memoriji ***MEM*** ne postoji razlika i da sada dati ulaz keš memorije ***KEŠ*** treba proglasiti za nevažeći. Zbog toga se u koraku $step_1$ generiše aktivna vrednost signala **clearV**, čime se u odgovarajući flip-flop V upisuje neaktivna vrednost, i potom prelazi iz koraka $step_1$ na korak $step_2$. U koraku $step_2$ se generišu aktivne vrednosti signala **CRP** i **clCSC**. Signalom **CRP** se procesoru ***CPU*** šalje indikacija da je operacija selektivnog vraćanja završena, a signalom **clCSC** se flip-flop CSC postavlja na neaktivnu vrednost. Iz koraka $step_2$ se prelazi na korak $step_0$. U koraku $step_0$ se čeka pojava aktivne vrednosti signala startovanja operacije čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja.

Vremenski oblici signala operacije selektivnog vraćanja za slučaj kada u keš memoriji nema saglasnosti dati su na slici 33.

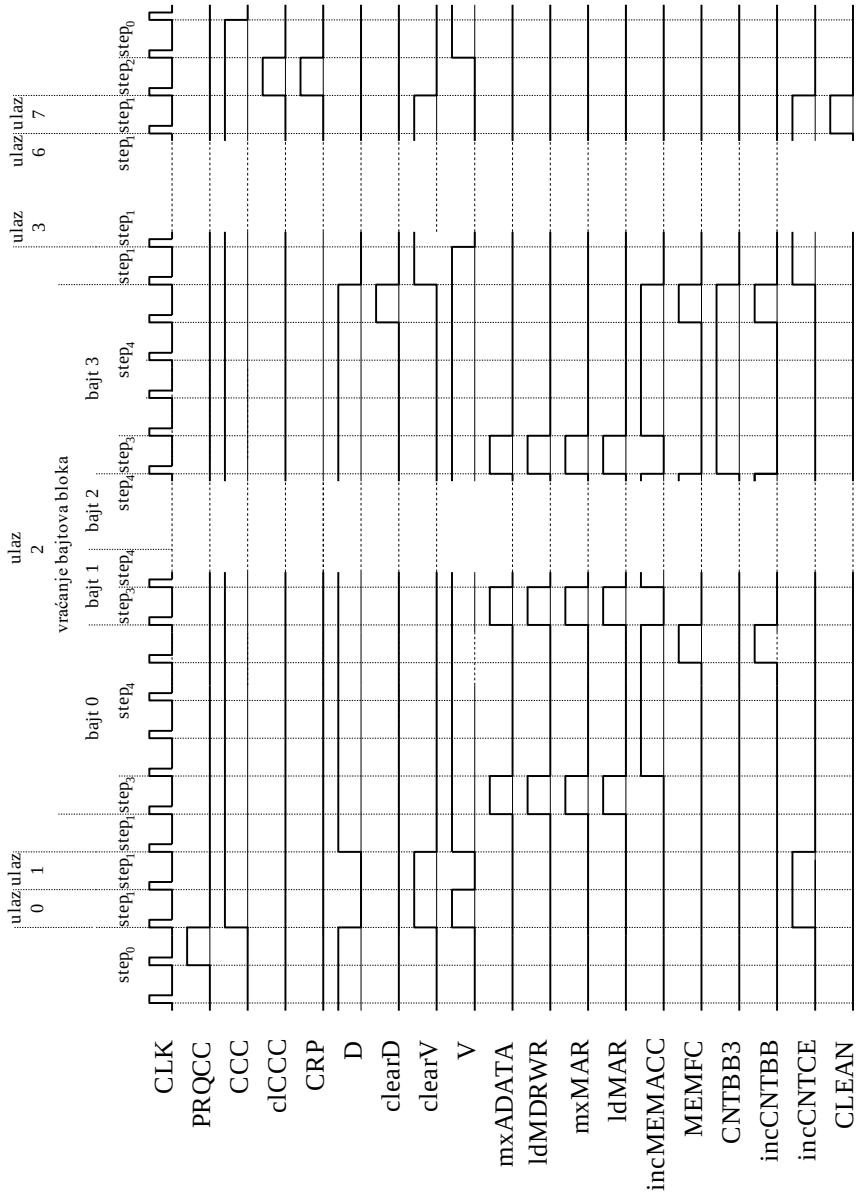


Slika 33 Vremenski oblici signala za slučaj operacija selektivno vraćanje i nema saglasnosti

Keš memorija ***KEŠ*** u koraku $step_0$ čeka pojavu aktivne vrednosti signala startovanja operacije selektivnog vraćanja **PRQSC**. Pri njenoj pojavi na signal takta se u registar CAR upisuje adresa memorije ***MEM*** za koju treba proveriti da li se blok podataka kome pripada i podatak sa date adrese nalazi u keš memoriji ***KEŠ***, flip-flop CSC se postavlja na aktivnu vrednost i prelazi se u korak $step_1$. U koraku $step_1$ se dobija da je signal **V** aktivan, što znači da je ulaz važeći, da je signal **H/M** neaktivan, što znači da se u datom ulazu keš memorije ***KEŠ*** ne nalazi blok iz grupe memorije ***MEM*** kojoj pripada i generisana adresa, i da je signal **D** aktivan, što znači da je blok iz datog ulaza keš memorije ***KEŠ*** modifikovan. Ova kombinacija signala **V**, **H/M** i **D** u slučaju operacije selektivnog vraćanja označava i da se u datom ulazu keš memorije ***KEŠ*** ne nalazi blok iz grupe memorije ***MEM*** kojoj pripada i generisana adresa i da sada ne treba ništa daditi sa datim ulazom keš memorije ***KEŠ***. Zbog toga u koraku ne generiše ni jedan signal već se samo prelazi iz koraka $step_1$ na korak $step_2$. U koraku $step_2$ se generišu aktivne vrednosti signala **CRP** i **clCSC**. Signalom **CRP** se procesoru ***CPU*** šalje indikacija da je operacija selektivnog vraćanja završena, a signalom **clCSC** se flip-flop CSC postavlja na neaktivnu vrednost. Iz koraka $step_2$ se prelazi na korak $step_0$. U koraku $step_0$ se čeka pojava aktivne vrednosti signala startovanja operacije čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja.

1.2.1.3.2.3.4. Operacija kompletnog vraćanja

Vremenski oblici signala operacije kompletnog vraćanja dati su na slici 34.



Slika 34 Vremenski oblici signala za slučaj operacija kompletno vraćanje

Keš memorija ***KEŠ*** u koraku $step_0$ čeka pojavu aktivne vrednosti signala startovanja operacije kompletnog vraćanja **PRQCC** koji preko bloka *cpu interfejs* dolazi iz procesora ***CPU***. Pri njenoj pojavi na signal takta flip-flop CCC bloka *cpu interfejs* se postavlja na aktivnu vrednost i prelazi se u korak $step_1$. U koraku $step_1$ se redom za ulaze 0 do 7 keš memorije ***KEŠ*** vrši provera vrednosti signala **V** i **D** bloka *indikatori*, pri čemu je broj ulaza određen vrednošću brojača CNTCE bloka *brojači*. Uzeto je da je za ulaz 0 keš memorije ***KEŠ*** signal **V** aktivan, što znači da je ulaz važeći, i da je signal **D** neaktivan, što znači da blok iz datog ulaza keš memorije ***KEŠ*** nije modifikovan. Ova kombinacija signala **V** i **D** u slučaju operacije kompletnog vraćanja označava da ne treba ništa raditi sa blokom iz datog ulaza keš memorije ***KEŠ***, već samo treba dati ulaz proglasiti za nevažeći i preći na sledeći ulaz keš memorije ***KEŠ***. Stoga se za ulaz 0 u koraku $step_1$ generišu aktivne vrednosti signale **clearV** bloka *indikatori* i **incCNTCE** bloka *brojači*, čime se na signal takta flip-flop V ulaza 0 postavlja na neaktivnu vrednost i brojač CNTCE inkrementira. U ovom slučaju se ostaje u koraku $step_1$, pri čemu se sada vrednošću 1 brojača CNTCE selektuju signali **V** i **D** ulaza 1 keš memorije ***KEŠ***. Uzeto je da je za ulaz 1 keš memorije ***KEŠ*** signal **V** neaktivan, što znači

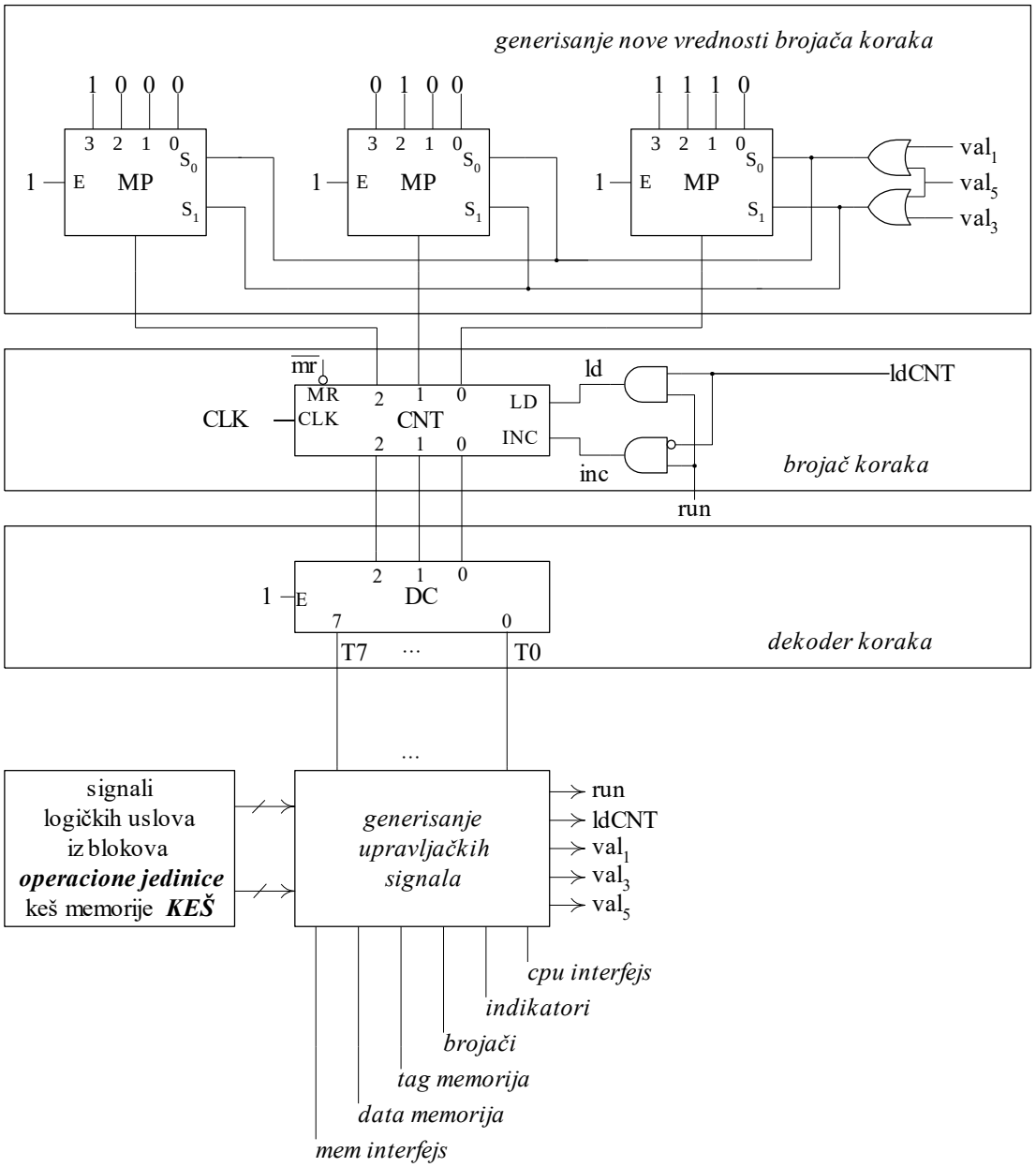
da je ulaz ne važeći, pa je i signal **D** neaktivan. Ova kombinacija signala **V** i **D** u slučaju operacije kompletnog vraćanja označava da ne treba ništa raditi sa blokom iz datog ulaza keš memorije ***KEŠ***, već samo treba preći na sledeći ulaz keš memorije ***KEŠ***. Stoga se za ulaz 1 u koraku $step_1$ generišu aktivne vrednosti signale **clearV** i **incCNTCE**, čime se na signal takta za flip-flop V ulaza 1 samo potvrđuje neaktivna vrednost i brojač CNTCE inkrementira. I u ovom slučaju se ostaje u koraku $step_1$, pri čemu se sada vrednošću 2 brojača CNTCE selektuju signali **V** i **D** ulaza 2 keš memorije ***KEŠ***. Uzeto je da je za ulaz 2 keš memorije ***KEŠ*** signal **V** aktivan, što znači da je ulaz važeći, i da je i signal **D** aktivan, što znači da je blok iz datog ulaza keš memorije ***KEŠ*** modifikovan. Zbog toga je potrebno, najpre, četiri puta proći kroz korake $step_3$ i $step_4$ radi vraćanja četiri bajta bloka iz keš memorije ***KEŠ*** u memoriju ***MEM***, i na kraju ponovo u koraku $step_1$ formirati vrednosti signala **V** i **D**. Vraćanja četiri bajta bloka iz keš memorije ***KEŠ*** u memoriju ***MEM*** se realizuje na identičan način kao i u slučaju operacija čitanja, nema saglasnosti, ulaz je važeći i blok je modifikovan. Zbog toga su sva objašnjenja vezana za aktivnosti i generisanje signala u koracima $step_3$ i $step_4$ i identična sa odgovarajućim objašnjenjima iz odeljka 1.2.1.3.2.3.1. Po završetku prebacivanje četiri bajta bloka iz keš memorije ***KEŠ*** u memoriju ***MEM*** prelazi se iz koraka $step_4$ na korak $step_1$. U koraku $step_1$ se sada dobija da je signal **V** aktivan, što znači da je ulaz važeći, i da je signal **D** neaktivan, što znači da blok iz datog ulaza keš memorije ***KEŠ*** nije modifikovan. Sada se postupa na identičan način kao i za ulaz 0 za koji je bila identična kombinacija signala **V** i **D**. Stoga se sada za ulaz 2 u koraku $step_1$ generišu aktivne vrednosti signale **clearV** i **incCNTCE**, čime se na signal takta flip-flop V ulaza 2 postavlja na neaktivnu vrednost i brojač CNTCE inkrementira. U ovom slučaju se ostaje u koraku $step_1$, pri čemu se sada vrednošću 3 brojača CNTCE selektuju signali **V** i **D** ulaza 3 keš memorije ***KEŠ***. Vremenski oblici signala za ulaze 3 do 6 nisu prikazani, već se prikazuju samo za ulaz 7. Ovaj ulaz je interesantan, jer je to zadnji ulaz keš memorije ***KEŠ*** i stoga i signal **CLEAN** bloka *brojači* postaje aktivan. Uzeto je da je za ulaz 7 keš memorije ***KEŠ*** signal **V** neaktivan, što znači da je ulaz ne važeći, pa je i signal **D** neaktivan. Ova kombinacija signala **V** i **D** u slučaju operacije kompletnog vraćanja označava da ne treba ništa raditi sa blokom iz datog ulaza keš memorije ***KEŠ***, već se samo treba vratiti na ulaz 0 keš memorije ***KEŠ***. Stoga se za ulaz 7 u koraku $step_1$ generišu aktivne vrednosti signale **clearV** i **incCNTCE**, čime se na signal takta za flip-flop V ulaza 7 samo potvrđuje neaktivna vrednost i brojač CNTCE inkrementiranjem vraća na početnu vrednost 0. Međutim, s obzirom da je signal **CLEAN** sada aktivan, u ovom slučaju se iz koraka $step_1$ prelazi na korak $step_2$. U koraku $step_2$ se generišu aktivne vrednosti signala **CRP** i **clCCC** bloka *cpu interfejs*. Signalom **CRP** se procesoru ***CPU*** šalje indikacija da je operacija kompletnog vraćanja završena, a signalom **clCCC** se flip-flop CCC postavlja na neaktivnu vrednost. Iz koraka $step_2$ se prelazi na korak $step_0$. U koraku $step_0$ se čeka pojava aktivne vrednosti signala startovanja operacije čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja.

1.2.1.3.2.4. Struktura upravljačke jedinice

Struktura ***upravljačke jedinice*** ožičene realizacije je prikazana na slici 35. ***Upravljačka jedinica*** 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.



Slika 35 Struktura upravljačke jedinice

1.2.1.3.2.4.1. Blok generisanje nove vrednosti brojača koraka

Blok *generisanje nove vrednosti brojača koraka* se sastoji od multipleksera i služi za generisanje i selekciju vrednosti koju treba upisati u brojač CNT. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog izvršavanja mikrooperacija. Analizom algoritma generisanja upravljačkih signala operacione jedinice (poglavlje 1.2.1.3.2.2) se utvrđuje da su 0, 1, 3 i 5 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 **val_1**, **val_3** i **val_5**. Ako su sva tri signala neaktivna kroz multipleksere se propušta vrednost 0, a aktivnom vrednošću samo jednog od signala **val_1**, **val_3** ili **val_5** kroz multiplekser se propuštaju vrednosti 1, 3 i 5, respektivno.

1.2.1.3.2.4.2. Blok brojač koraka

Blok *brojač koraka* se sastoji od brojača CNT koji 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 sekvence upravljačkih signala po koracima. Režim inkrementiranja je najčešći režim rada brojača CNT. 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 vrši se upis nove vrednosti u brojač CNT. 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 ne menja se vrednost brojača CNT. 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 startovanja operacije čitanja **CRQRD**, operacije upisa **CRQWR**, operacije selektivnog vraćanja **CRQSC** ili operacije kompletnog vraćanja **CRQCC** ili
- pojava signala **MEMFC** kojim se označava da je čitanje ili upis u memoriju MEM obavljen.

1.2.1.3.2.4.3. Blok dekoder koraka

Blok *dekoder koraka* se sastoji od dekodera DC. Na ulaze dekodera DC dovede se signali sa izlaza brojača CNT bloka *brojač koraka*. Dekodovana stanja brojača CNT pojavljuju se kao signali **T0** do **T7** na izlazima dekodera DC. Svakom koraku iz algoritma generisanja upravljačkih signala (poglavlje 1.2.1.3.2.2) dodeljen je jedan od ovih signala i to koraku $step_0$ signal **T0**, koraku $step_1$ signal **T1**, itd.

1.2.1.3.2.4.4. Blok generisanje upravljačkih signala

Blok *generisanje upravljačkih signala* se sastoji od kombinacionih mreža koje pomoću signala **T0** do **T7** koji dolaze iz bloka *dekoder koraka*, signala logičkih uslova koji dolaze iz blokova ***operacione jedinice*** i saglasno algoritmu generisanja upravljačkih signala (poglavlje 1.2.1.3.2.2) generišu upravljačke signale. Postupak projektovanja kombinacionih mreža je identičan sa postupkom koji je objašnjen u poglavlju 1.2.1.1.2.4.4.

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

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

1.2.1.3.2.4.4.1. Upravljački signali operacione jedinice

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

- blok *cpu interfejs*,
- blok *indikatori*,
- blok *brojači*,
- blok *tag memorija*,
- blok *data memorija* i
- blok *mem interfejs*.

1.2.1.3.2.4.4.1.1. Blok cpu interfejs

Upravljački signali koji odgovaraju ovom bloku ***operacione jedinice*** su:

- **ldCDRRD** — signal paralelnog upisa u registar CDRRD,
- **clCRD** — signal upisa neaktivne vrednosti u flip-flop CRD,
- **clCWR** — signal upisa neaktivne vrednosti u flip-flop CRD,
- **clCSC** — signal upisa neaktivne vrednosti u flip-flop CSC,
- **clCCC** — signal upisa neaktivne vrednosti u flip-flop CCC i
- **CRP** — signal kompletiranja operacija čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja.

Ovi upravljački signali se generišu na sledeći način:

- **ldCDRRD** = **T1** · **CRD**·**H/M**,
- **clCRD** = **T2** · **CRD**,
- **clCWR** = **T2** · **CWR**,
- **clCSC = T2** · **CSC**,
- **clCCC = T2** · **CCC** i
- **CRP** = **T2**.

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

- **H/M** — blok *tag memorija*,
- **CRD** —blok *cpu interfejs*,
- **CWR** — blok *cpu interfejs*,
- **CSC** — blok *cpu interfejs* i
- **CCC** — blok *cpu interfejs*.

1.2.1.3.2.4.4.1.2. Blok indikatori

Upravljački signali koji odgovaraju ovom bloku ***operacione jedinice*** su:

- **writeV** — signal upisa aktivne vrednosti u adresirani flip-flop V,
- **clearV** — signal upisa neaktivne vrednosti u adresirani flip-flop V,
- **writeD** — signal upisa aktivne vrednosti u adresirani flip-flop D i
- **clearD** — signal upisa neaktivne vrednosti u adresirani flip-flop D.

Ovi upravljački signali se generišu na sledeći način:

- **writeV** = **T7** · **CNTBB3**,
- **clearV** = **T1** · (**CSC** · **H/M** + **CCC**) · ,
- **writeD** = **T2** · **CWR** i
- **clearD** = **T4** · (**CSC** + **CCC**) · **CNTBB3** · **MEMFC** + **T7** · **CNTBB3**.

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

- **CNTBB3** — blok *brojači*,
- **CWR** — blok *cpu interfejs*,
- **CSC** — blok *cpu interfejs*,
- **CCC** — blok *cpu interfejs*,
- □ — blok *indikatori*,
- **H/M** — blok *tag memorija* i
- **MEMFC** — blok *brojači*.

1.2.1.3.2.4.4.1.3. Blok brojači

Upravljački signali koji odgovaraju ovom bloku ***operacione jedinice*** su:

- **incCNTBB** — signal inkrementiranja sadržaja brojača CNTBB,
- **incMEMACC** — signal inkrementiranja sadržaja brojača MEMACC i
- **incCNTCE** — signal inkrementiranja sadržaja brojača CNTCE.

Ovi upravljački signali se generišu na sledeći način:

- **incCNTBB** = **T4** · **MEMFC** + **T7**,
- **incMEMACC** = **T4** + **T6** i
- **incCNTCE** = **T1** · **CCC**·□.

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

- **MEMFC** — blok *brojači*,
- **CCC** — blok *cpu interfejs* i
- □ — blok *indikatori*.

1.2.1.3.2.4.4.1.4. Blok tag memorija

Upravljački signal koji odgovara ovom bloku ***operacione jedinice*** je:

- **writeTAG** — signal upisa u memorijski modul TAG.

Ovaj upravljački signal se generišu na sledeći način:

- **writeTAG** = **T7** · **CNTBB3**.

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

- **CNTBB3** — blok *brojači*.

1.2.1.3.2.4.4.1.5. Blok data memorija

Upravljački signali koji odgovaraju ovom bloku ***operacione jedinice*** su:

- **mxDIDATA** — signal selekcije kroz multiplekser,
- **mxADATA** — signal selekcije kroz multiplekser i
- **writeDATA** — signal upisa u memorijski modul DATA.

Ovi upravljački signali se generišu na sledeći način:

- **mxDIDATA** = **T7**,
- **mxADATA** = **T3** + **T7** i
- **writeDATA** = **T2** · **CWR** + **T7**.

Pri njihovom generisanju koristi se signal logičkog uslova koji dolazi iz bloka ***operacione jedinice*** i to:

- **CWR** — blok *cpu interfejs*.

1.2.1.3.2.4.4.1.6. Blok mem interfejs

Upravljački signali koji odgovaraju ovom bloku ***operacione jedinice*** su:

- **mxMAR** — signal selekcije kroz multiplekser,
- **ldMAR** — signal paralelnog upisa u registar MAR,
- **ldMDRWR** — signal paralelnog upisa u registar MDRWR,
- **ldMDRRD** — signal paralelnog upisa u registar MDRRD i
- **writeMEM** — signal upisa u memoriju ***MEM***.

Ovi upravljački signali se generišu na sledeći način:

- **mxMAR** = **T3**,
- **ldMAR** = **T3** + **T5**,
- **ldMDRWR** = **T3**,
- **ldMDRRD** = **T6** · **MEMFC** i
- **writeMEM** = **T4**.

Pri njihovom generisanju koristi se signal logičkog uslova koji dolazi iz bloka ***operacione jedinice*** i to:

- **MEMFC** — blok *brojači*.

1.2.1.3.2.4.4.2. Upravljački signali upravljačke jedinice

Upravljački signali upravljačke jedinice su:

- **ldCNT** — signal čijom se aktivnom vrednošću, pod uslovom da je signal **run** aktivan, upisuje jedna od vrednosti 0, 1, 3 ili 5 u brojač CNT, a neaktivnom vrednošću, pod uslovom da je signal **run** aktivan, inkrementira tekuća vrednosti brojača CNT,
- **run** — signal čijom se aktivnom vrednošću omogućuje, u zavisnosti od vrednosti signala **ldCNT,** ili upis nove vrednosti ili inkrementiranje vrednosti brojača CNT, a neaktivnom vrednošću, bez obzira na vrednost signala **ldCNT**, onemogućava promena vrednosti brojača CNT i
- $\mathbf{val_1}$, $\mathbf{val_3}$ i $\mathbf{val_5}$ — signali koji obezbeđuju generisanje vrednosti 0, 1, 3 i 5 za upis u brojač CNT.

Ovi upravljački signali se generišu na sledeći način:

- $\mathbf{ldCNT} = \mathbf{T1} \cdot ((\mathbf{CRD} + \mathbf{CWR}) \cdot \overline{\mathbf{H/M}} + \mathbf{CSC} \cdot \mathbf{H/M} \cdot \mathbf{D} + \mathbf{CCC} \cdot \mathbf{D}) + \mathbf{T2} + \mathbf{T4} \cdot (\overline{\mathbf{CNTBB3}} + \mathbf{CNTBB3} \cdot (\mathbf{CSC} + \mathbf{CCC})) \cdot \mathbf{MEMFC} + \mathbf{T7}$,
- $\mathbf{run} = \mathbf{T0} \cdot (\mathbf{PRQRD} + \mathbf{PRQWR} + \mathbf{PRQSC} + \mathbf{PRQCC}) + \mathbf{T1} \cdot ((\mathbf{CRD} + \mathbf{CWR} + \mathbf{CSC}) + \mathbf{CCC} \cdot (\mathbf{D} + \overline{\mathbf{D}} \cdot \mathbf{CLEAN}) + \mathbf{T2} + \mathbf{T3} + \mathbf{T4} \cdot \mathbf{MEMFC} + \mathbf{T5} + \mathbf{T6} \cdot \mathbf{MEMFC} + \mathbf{T7}$,
- $\mathbf{val_1} = \mathbf{T4} \cdot (\mathbf{CSC} + \mathbf{CCC}) \cdot \mathbf{CNTBB3} \cdot \mathbf{MEMFC} + \mathbf{T7} \cdot \mathbf{CNTBB3}$,
- $\mathbf{val_3} = \mathbf{T1} \cdot ((\mathbf{CRD} + \mathbf{CWR}) \cdot \overline{\mathbf{H/M}} + \mathbf{CSC} \cdot \mathbf{H/M} + \mathbf{CCC}) \cdot \mathbf{D} + \mathbf{T4} \cdot \overline{\mathbf{CNTBB3}} \cdot \mathbf{MEMFC}$ i
- $\mathbf{val_5} = \mathbf{T1} \cdot (\mathbf{CRD} + \mathbf{CWR}) \cdot \overline{\mathbf{H/M}} \cdot \overline{\mathbf{D}} + \mathbf{T7} \cdot \overline{\mathbf{CNTBB3}}$.

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

- $\overline{\textbf{H/M}}$ — blok *tag memorija,*
- **CNTBB3** — blok *brojači,*
- **PRQRD** — blok *cpu interfejs,*
- **PRQWR** — blok *cpu interfejs,*
- **PRQSC** — blok *cpu interfejs,*
- **PRQCC** — blok *cpu interfejs,*
- **MEMFC** — blok *brojači,*
- **CLEAN** — blok *brojači,*
- **D** — blok *indikatori,*
- **CRD** — blok *cpu interfejs,*
- **CWR** — blok *cpu interfejs,*
- **CSC** — blok *cpu interfejs* i
- **CCC** — blok *cpu interfejs*.

1.2.2. KEŠ MEMORIJA SA ASOCIJATIVNIM PRESLIKAVANJEM

Keš memorija koja se razmatra realizovana je tehnikom asocijativnog preslikavanja na nivou bloka veličine 4 bajta. U ovoj tehnici preslikavanja bilo koji blok operativne memorije može da se smesti u bilo koji ulaz keš memorije, pa je stoga neophodno koristiti neki od algoritama zamene. U usvojenoj realizaciji koristi se LRU algoritam zamene. Za ažuriranje operativne memorije koristi se tehnika *upiši-skroz* sa baferovanjem. Radi smanjivanja čekanja procesora koristi se tehnika *by-pass*. Usvojeno je da se kod operacije upisa podatak upisuje uvek u operativnu memoriju, dok se u keš memoriju upisuje samo u slučaju saglasnosti. Blok podataka sa dovlači iz operativne memorije u keš memoriju samo u slučaju da kod operacije čitanja nije otkrivena saglasnost. U slučaju da kod operacije upisa nije otkrivena saglasnost, nema dovlačenja bloka podataka iz operativne memorije u keš memoriju, već se upis vrši samo u operativnu memoriju.

Keš memorija ima 8 ulaza u kojima se čuva 8 blokova operativne morije. Kapacitet dela keš memorije u kome se čuva sadržaj je 32 bajta, a adresabilna jedinica je jedan bajt.

Operativna memorija je kapaciteta 64KB, a adresabilna jedinica je jedan bajt. Stoga se operativna memorija može posmatrati kao da je organizovana u 2^{14} blokova veličine 2^2 bajta. Adresa operativne memorije dužine 16 bita može se podeliti i označiti na sledeći način:

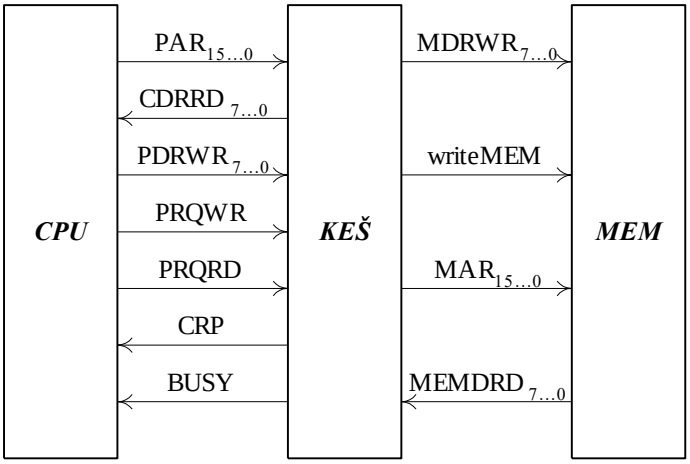
- viših 14 bitova označavaju broj bloka i
- niža 2 bita označavaju adresu bajta u bloku.

Keš memorija je deo sistema (slika 36) koji se sastoji iz:

- procesora ***CPU***,
- memorije ***MEM*** i
- keš memorije ***KEŠ***.

Uzeto je da su sve sekvencijalne mreže sistema sinhrone i da ceo sistem radi sinhrono sa zajedničkim signalom takta **CLK**.

U ovom poglavlju daju se detalji realizacije procesora ***CPU*** i memorije ***MEM*** relevantni za rad keš memorije ***KEŠ*** i kompletna hardverska realizacija keš memorije ***KEŠ***.

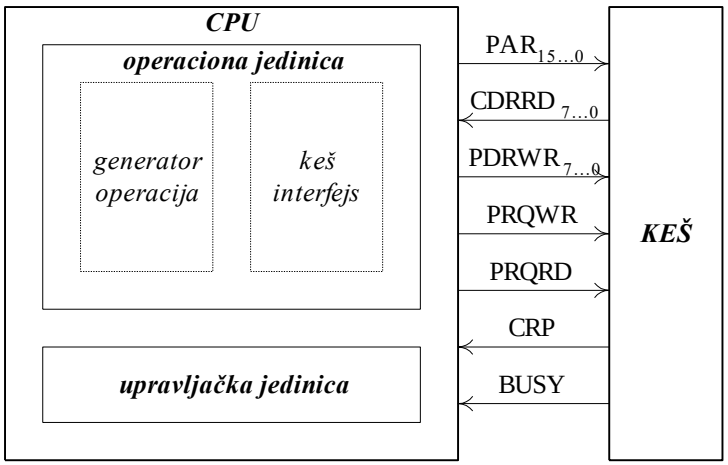


Slika 36 Struktura sistema

1.2.2.1. PROCESOR CPU

Procesor ***CPU*** se obraća keš memoriji ***KEŠ*** onda kada treba očitati podatak iz keš memorije ***KEŠ*** ili upisati podatak u keš memoriju ***KEŠ***. Ova dva obraćanja procesora ***CPU*** keš memoriji ***KEŠ*** se u daljem tekstu nazivaju operacija čitanja i operacija upisa. Operacije čitanja ili upisa se koriste kad god se generiše adresa memorije ***MEM*** sa koje treba očitati ili instrukciju ili operand ili na kojoj treba upisati rezultat.

Signali koji se razmenjuju između procesora ***CPU*** i keš memorije ***KEŠ*** prilikom izvršavanja ove dve operacije prikazani su na slici 37 .



Slika 37 Procesor CPU

Kod operacije čitanja procesor ***CPU*** šalje keš memoriji ***KEŠ*** 16-bitnu adresu po linijama **$PAR_{15...0}$** i generiše aktivnu vrednost signala **PRQRD** trajanja jedna perioda signala takta. Keš memorija ***KEŠ*** vraća očitani 8-bitni podatak po linijama **$CDRRD_{7...0}$**. Keš memorija ***KEŠ*** šalje procesoru ***CPU*** i signale **CRP** i **BUSY**. Aktivnom vrednošću signala **CRP** trajanja jedna perioda signala takta keš memorija ***KEŠ*** signalizira procesoru ***CPU*** da je na linijama **$CDRRD_{7...0}$** važeći podatak i da procesor ***CPU*** može da produži sa radom. Aktivnom i neaktivnom vrednošću signal **BUSY** keš memorija ***KEŠ*** signalizira procesoru ***CPU*** da li je zauzeta ili ne, respektivno. Prilikom startovanja operacije čitanja aktivnom vrednošću signala **PRQRD** keš memorija ***KEŠ*** postavlja signal **BUSY** na aktivnu vrednost. U slučaju da u keš memoriji ***KEŠ*** ima saglasnosti očitani podatak se šalje u procesor ***CPU***, generiše aktivna

vrednost signala **CRP** i signal **BUSY** postavlja na neaktivnu vrednost. Time se procesoru ***CPU*** omogućava produži sa radom i eventualno startuje novu operaciju u keš memoriji ***KEŠ***. U slučaju da u keš memoriji ***KEŠ*** nema saglasnosti iz memorije ***MEM*** se u keš memoriju ***KEŠ*** dovlači najpre bajt podatka sa generisane adrese, očitani podatak odmah šalje u procesor ***CPU*** i generiše aktivna vrednost signala **CRP**, čime se procesoru dozvoljava da produži sa radom. Međutim, signal **BUSY** ostaje aktivan sve dok je keš memorija ***KEŠ*** zauzeta dovlačenjem preostalih bajtova bloka, čime se procesoru ***CPU*** onemogućava da startuje novu operaciju u keš memoriji ***KEŠ*** pre kompletiranja prenosa svih bajtova bloka iz memorije ***MEM*** u keš mamoriju ***KEŠ.***

Kod operacije upisa procesor ***CPU*** šalje keš memoriji ***KEŠ*** 16-bitnu adresu po linijama $\mathbf{PAR_{15...0}}$, 8-bitni podatak za upis po linijama $\mathbf{PDRWR_{7...0}}$ i generiše aktivnu vrednost signala **PRQWR** trajanja jedna perioda signala takta. Keš memorija ***KEŠ*** šalje procesoru ***CPU*** signale **CRP** i **BUSY**. Aktivnom vrednošću signala **CRP** trajanja jedna perioda signala takta keš memorija ***KEŠ*** signalizira procesoru ***CPU*** da procesor ***CPU*** može da produži sa radom. Aktivnom i neaktivnom vrednošću signal **BUSY** keš memorija ***KEŠ*** signalizira procesoru ***CPU*** da li je zauzeta ili ne, respektivno. Prilikom startovanja operacije upisa aktivnom vrednošću signala **PRQWR** keš memorija ***KEŠ*** upisuje podatak sa linija $\mathbf{PDRWR_{7...0}}$ u prihvatni registar podatka, generiše aktivna vrednost signala **CRP** i signal **BUSY** postavlja na aktivnu vrednost. Time se procesoru ***CPU*** omogućava da produži sa radom. Međutim, signal **BUSY** je aktivan sve dok je keš memorija ***KEŠ*** zauzeta upisom podatka iz prihvatnog registra podatka u memoriju ***MEM***, a u slučaju saglasnosti i u keš memoriju ***KEŠ***, čime se procesoru ***CPU*** onemogućava da startuje novu operaciju u keš memoriji ***KEŠ*** pre kompletiranja upisa**.**

Procesor ***CPU*** (slika 37) 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 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 prema algoritmu generisanja upravljačkih signala operacija procesora ***CPU*** i signala logičkih uslova.

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

1.2.2.1.1. Operaciona jedinica

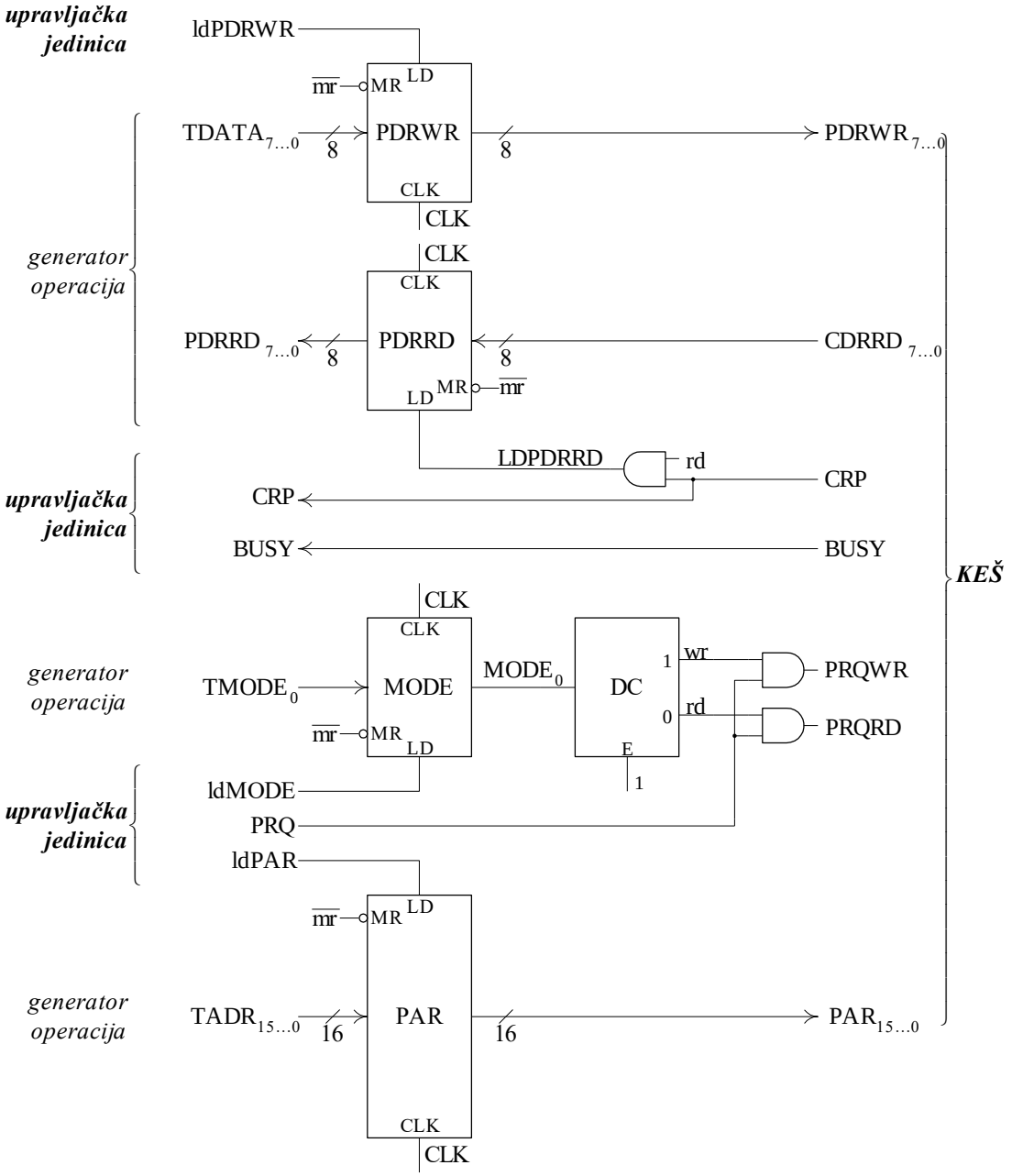
Operaciona jedinica (slika 37) se sastoji iz sledećih blokova:

- blok *keš interfejs* i
- blok *generator operacija*.

Blok *keš interfejs* služi za povezivanje procesora ***CPU*** i keš memorije ***KEŠ***. Blok *generator operacija* služi za generisanje operacija čitanja i upisa. Struktura i opis blokova ***operacione jedinice*** se daju u daljem tekstu.

1.2.2.1.1.1. Blok keš interfejs

Blok *keš interfejs* (slika 38) sadrži registre PAR, PDRWR, PDRRD i MODE, dekoder DC i dva logička I kola.



Slika 38 Blok keš interfejs

Registar PAR (Processor Address Register), PDRWR (Processor Data Register for WRite) i PDRRD (Processor Data Register for ReaD) imaju identičnu funkciju kao i registri sa identičnim nazivima u odeljku 1.2.1.1.1.1 .

Registar MODE (operation MODE) služi za čuvanje binarne vrednosti jedne od dve operacije koje keš memorija ***KEŠ*** može da realizuje. Pretpostavlja se da je procesor ***CPU*** pre obraćanja keš memoriji ***KEŠ*** već upisao binarnu vrednost operacije u registar MODE, tako što je generisao aktivnu vrednost signala **ldMODE** trajanja jedna perioda signala takta i najkasnije signal takta kojim se signal **PRQ** (Processor ReQuest) postavlja na aktivnu vrednost izvršio upis sadržaja sa linija $\mathbf{TMODE_0}$ u registar MODE. Signal $\mathbf{MODE_0}$ sa izlaza registra MODE se vodi na ulaze dekodera DC.

Dekoder DC i dva logička I kola služe za generisanje signala **PRQRD** i **PRQWR**. Dekoder DC na svojim izlazima daje aktivnu vrednost jednog od signala **rd** i **wr** u zavisnosti

od binarne vrednosti signala $\mathbf{MODE_0}$ sa ulaza. Pri aktivnoj vrednosti signala **PRQ** trajanja jedna perioda signala takta jedan od signala **PRQRD** i **PRQWR** postaje aktivan u zavisnosti od toga koji je od signala **rd** i **wr** aktivan. Signali **PRQRD** i **PRQWR** su neaktivni pri neaktivnoj vrednosti signala **PRQ**.

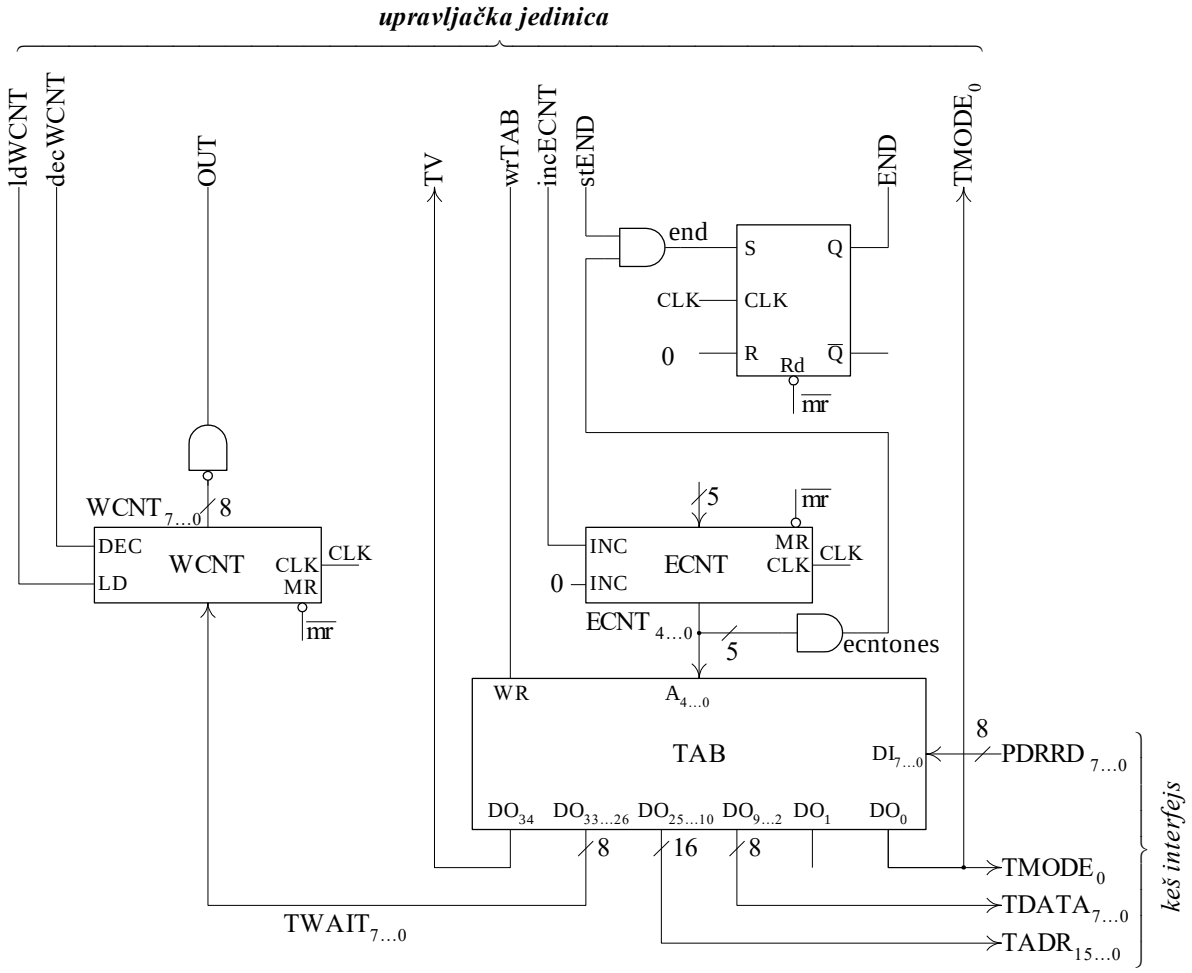
Upravljački signali **PRQRD** (Processor ReQuest for ReaD) i **PRQWR** (Processor ReQuest for WRite) imaju identičnu funkciju kao upravljački signali sa identičnim nazivima u odeljku 1.2.1.1.1.1.

Upravljački signal **CRP** (Cache RePly) se koristi da keš memorija ***KEŠ*** aktivnom vrednošću ovog signala trajanja jedna perioda signala takta signalizira procesoru ***CPU*** da može da produži sa radom iako operacija čitanja ili upisa nije završena. U slučaju operacije čitanja aktivnom vrednošću signala **CRP** se signalizira i da se na linijama $\mathbf{CDRRD_{7...0}}$ nalazi očitani podatak i da aktivnom vrednošću signala **CRP** treba na signal takta sadržaj sa linija $\mathbf{CDRRD_{7...0}}$ upisati u registar PDRRD.

Upravljački signal **BUSY** se koristi da keš memorija ***KEŠ*** aktivnom vrednošću ovog signala signalizira procesoru ***CPU*** da je još uvek zauzeta jer sve aktivnosti u keš memoriji ***KEŠ*** vezane za operaciju čitanja ili upisa nisu završene i da procesor ***CPU*** ne sme da startuje novu operaciju čitanja ili upisa.

1.2.2.1.1.2. Blok generator operacija

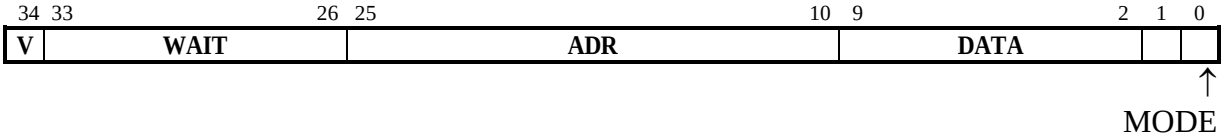
Blok *generator operacija* (slika 39) sadrži brojače ECNT i WCNT, flip-flop END i memoriju TAB.



Slika 39 Generator operacija

Brojači ECNT (Entry CouNTer) i WCNT (Wait CouNTer) i flip-flop END (generation of operations ENDed) imaju identičnu funkciju kao i registri i flip-flop sa identičnim nazivima u odeljku 1.2.1.1.1.2.

Memorija TAB ima 32 ulaza, pa najviše 32 operacije keš memorije ***KEŠ*** mogu da budu generisane. Jedan ulaz sadrži informacije na osnovu kojih se generiše jedna operacija keš memorije ***KEŠ*** (slika 40).



Slika 40 Struktura jednog ulaza u memoriji TAB

Polja V (Valid) i WAIT (WAIT between operations) imaju identičnu funkciju kao i polja sa identičnim nazivima u odeljku 1.2.1.1.1.2

Polje ADR (ADdRess) označava adresu memorije ***MEM*** sa koje treba očitati podatak u slučaju operacije čitanja i adresu memorije ***MEM*** na kojoj treba upisati podatak u slučaju operacije upisa. Polje DATA (DATA to be written or read) predstavlja podatak koji se upisuje u memorije ***MEM*** u slučaju operacije upisa i podatak koji je očitan iz memorije ***MEM*** u

slučaju operacije čitanja. Ova polja imaju identičnu funkciju kao i polja sa identičnim nazivima u odeljku 1.2.1.1.1.2.

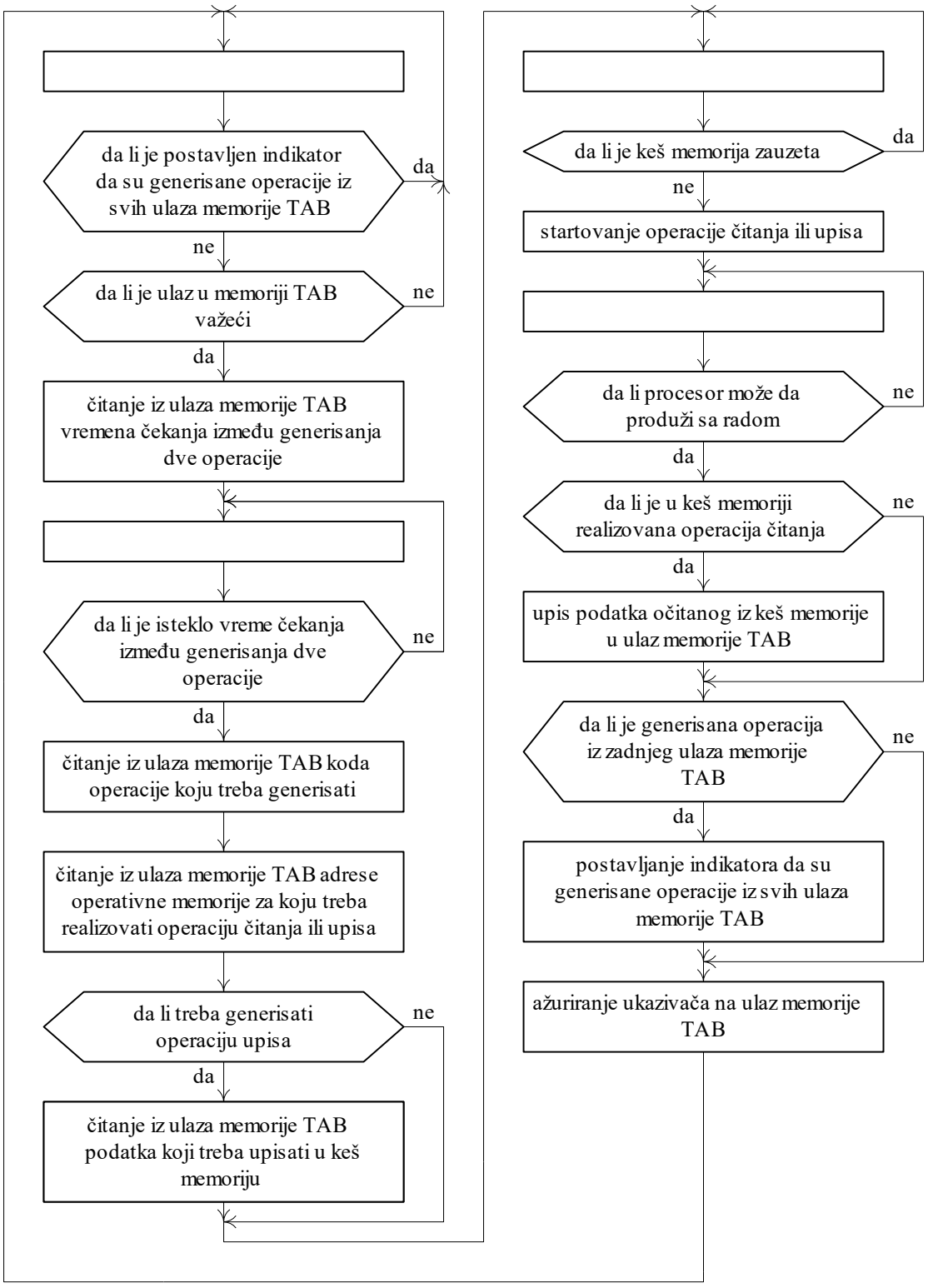
Polje MODE (MODE of operation) predstavlja kod za operaciju koju treba da realizuje keš memorija ***KEŠ***. Vrednostima od 0 i 1 označene su operacija čitanja i operacija upisa, respektivno. Bit MODE se pojavljuje na liniji DO_0 i vodi se kao signal $\mathbf{TMODE_0}$ u blok *keš interfejs*.

1.2.2.1.2. Upravljačka jedinica

U ovom poglavlju se prikazuju dijagram toka generisanja operacija, algoritam generisanja upravljačkih signala, vremenski oblici signala i struktura ***upravljačke jedinice***.

1.2.2.1.2.1. Dijagram toka generisanja operacija

Dijagram toka generisanja operacija čitanja i upisa je dat na slici 41.



Slika 41 Dijagram toka generisanja operacija

U početnom koraku se vrši provera da li je postavljen indikator da su generisane operacije iz svih ulaza memorije TAB. Ukoliko je indikator postavljen ostaje se u početnom koraku i nema generisanja operacija. 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 operacija. Ukoliko je ulaz važeći prelazi se na korake generisanja operacije čitanja ili upisa. Iz ulaza memorije TAB se, najpre, čita vreme čekanja između generisanja dve operacije, a po isteku ovog vremena i kod

operacije koju treba generisati. Potom se iz ulaza memorije TAB čita i adresa operativne memorije za koju treba operaciju čitanja ili upisa realizovati, a u slučaju operacije upisa čita se i podatak koji treba upisati u keš memoriju. Potom se vrši provera da li je keš memorija zauzeta. Sve dok je keš memorija zauzeta nema startovanja nove operacije u keš memoriji. Tek kada keš memorija nije zauzeta odgovarajuća operacija se startuje u keš memoriji. Po prijemu indikacije da procesor može da produži sa radom procesor produžava sa preostalim koracima. Najpre se u slučaju da je u keš memoriji startovana operacija čitanja podatak očitan iz keš memorije upisuje u ulaz memorije TAB. Potom se u slučaju da je generisana operacija bila operacija generisana iz zadnjeg ulaza memorije TAB postavlja indikator da su generisane operacije iz svih ulaza memorije TAB. Na kraju se vrši ažuriranje ukazivača na ulaz memorije TAB i prelazi na početni korak.

1.2.2.1.2.2. Algoritam generisanja upravljačkih signala

Algoritam generisanja upravljačkih signala je formiran na osnovu strukture ***operacione jedinice*** (poglavlje 1.2.2.1.1) i dijagrama toka generisanja operacija čitanja i upisa (poglavlje 1.2.2.1.2.1). Notacija koja se koristi je identična sa notacijom iz poglavlja 1.2.1.1.2.2.

Algoritam generisanja upravljačkih signala je dat u daljem tekstu.

$step_0$: *if* ((**TV** · $\overline{\textbf{END}}$), **ldWCNT**),

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

! Od koraka $step_0$ se kreće prilikom generisanja svake nove operacije. U korak $step_0$ se dolazi iz koraka $step_6$ po kompletiranju generisanja zadnje operacije. U koraku $step_0$ se proverava da li signali **TV** i $\overline{\textbf{END}}$ bloka *generisanje operacija* imaju aktivne vrednosti što se dešava u slučaju kada je odgovarajući ulaz memorije TAB bloka *generisanje operacija* važeći i kada još uvek nije generisana operacija za zadnji ulaz memorije TAB, respektivno. U slučaju da signali **TV** i $\overline{\textbf{END}}$ imaju aktivne vrednosti generiše se aktivna vrednost signala **ldWCNT** bloka *generisanje operacija* trajanja jedna perioda signala takta kojom se u brojač WCNT upisuje vrednost iz polja WAIT adresiranog ulaza memorije TAB i prelazi na korak $step_1$. U suprotnom slučaju ostaje se u koraku $step_0$.

$step_1$: *if* ($\overline{\textbf{OUT}}$, **decWCNT**),

if (**OUT**, **ldMODE**),

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 *generisanje operacija*. Neaktivna vrednost signala **OUT** označava da brojač WCNT još uvek nije došao do nule i da stoga treba generisati signal **decWCNT** bloka *generisanje operacija* i ostati u koraku $step_1$. Aktivna vrednost signala **OUT** označava da je brojač WCNT kao rezultat dekrementiranja došao do nule i da stoga treba generisati aktivnu vrednost signala **ldWCNT** trajanja jedna perioda signala takta kojom se u brojač WCNT upisuje vrednost iz polja MODE adresiranog ulaza memorije TAB bloka *generisanje operacija* i preći na korak $step_2$.

$step_2$: **ldPAR**,

if (**$TMODE_0$, ldPDRWR**),

br $step_3$

! U korak $step_2$ se dolazi iz koraka $step_1$. U ovom koraku se bezuslovno generiše aktivna vrednost signala **ldPAR** trajanja jedna perioda signala takta kojom se u registar PAR bloka *keš interfejs* upisuje vrednost iz polja ADR adresiranog ulaza memorije TAB bloka *generisanje operacija*. Ukoliko je aktivna vrednost signala **$TMODE_0$** radi se o operaciji upisa, pa se generiše aktivna vrednost signala **ldPDRWR** trajanja jedna perioda signala takta kojom se u registar PDRWR bloka *keš interfejs* upisuje vrednost iz polja DATA adresiranog ulaza memorije TAB. U koraku $step_2$ se ostaje samo jedna perioda signala takta i bezuslovno se prelazi u korak $step_3$.

$step_3$: *if* ($\overline{\textbf{BUSY}}$, **PRQ**),

br (*if* $\overline{\textbf{BUSY}}$ *then* $step_4$ *else* $step_3$)

! U korak $step_3$ se dolazi iz koraka $step_2$. U ovom koraku se generiše aktivna vrednost signala **PRQ** bloka *keš interfejs* trajanja jedna perioda signala takta ukoliko je neaktivan signal **BUSY** koji preko bloka *keš interfejs* keš memorija ***KEŠ*** šalje procesoru ***CPU***. Time se u zavisnosti od sadržaja registra MODE bloka *keš interfejs* generiše aktivna vrednost signala **PRQRD** ili **PRQWR** i startuje keš memorija ***KEŠ*** da izvrši operaciju čitanja ili upisa, respektivno. U koraku $step_3$ se ostaje jedna ili više perioda signala takta u zavisnosti od toga da li je pri prelasku na korak $step_3$ signal **BUSY** bio neaktivan ili aktivan, respektivno. Pri neaktivnoj vrednosti signal **BUSY** se prelazi u korak $step_4$.

$step_4$: *br* (*if* **CRP** *then* $step_5$ else $step_4$)

! U korak $step_4$ se dolazi iz koraka $step_3$. U koraku $step_4$ se vrši provera signala **CRP** koji preko bloka *keš interfejs* keš memorija ***KEŠ*** šalje procesoru ***CPU***. Neaktivna vrednost signala **CRP** je indikacija da procesor ***CPU*** mora da čeka na keš memoriju ***KEŠ***, dok je **a**ktivna vrednost signala **CRP** indikacija da procesor ***CPU*** može da produži sa radom. U sučaju operacije čitanja, aktivnom vrednošću signala **CRP** trajanja jedna perioda signala takta se procesoru **CPU** šalje i indikacija da je očitani bajt podatka raspoloživ na linijama CDRRD. Kako je za operaciju čitanja signal **rd** aktivan, pojavljivanjem aktivne vrednosti signala **CRP** i signal LDPDRRD postaje aktivan, čime se omogućava upisivanje očitanog bajta podatka u registar PDRRD bloka *keš interfejs*. U koraku $step_4$ se ostaje sve dok je neaktivna vrednost signal **CRP**. Kada signal **CRP** postane aktivan prelazi se na korak $step_5$.

$step_5$: *if* ($\overline{\mathbf{TMODE_0}}$ **, wrTAB**),
br $step_6$

! U korak $step_5$ se dolazi iz koraka $step_4$. U ovom koraku se u slučaju operacije čitanja, kada je signal $\mathbf{TMODE_0}$ bloka *generisanje operacija* neaktivan, generiše aktivna vrednost signala **wrTAB** bloka *generisanje operacija* trajanja jedna perioda signala takta. Signalom **wrTAB** se očitani podatak upisuje iz registra PDRRD bloka *keš interfejs* u polje DATA adresiranog ulaza memorije TAB bloka *generisanje operacija*. U slučaju operacije upisa u ovom koraku se ne generiše ni jedan od upravljačkih signala. U koraku $step_5$ se ostaje samo jedna perioda signala takta i bezuslovno se prelazi u korak $step_6$.

$step_6$: **incECNT**, **stEND**,
br $step_0$

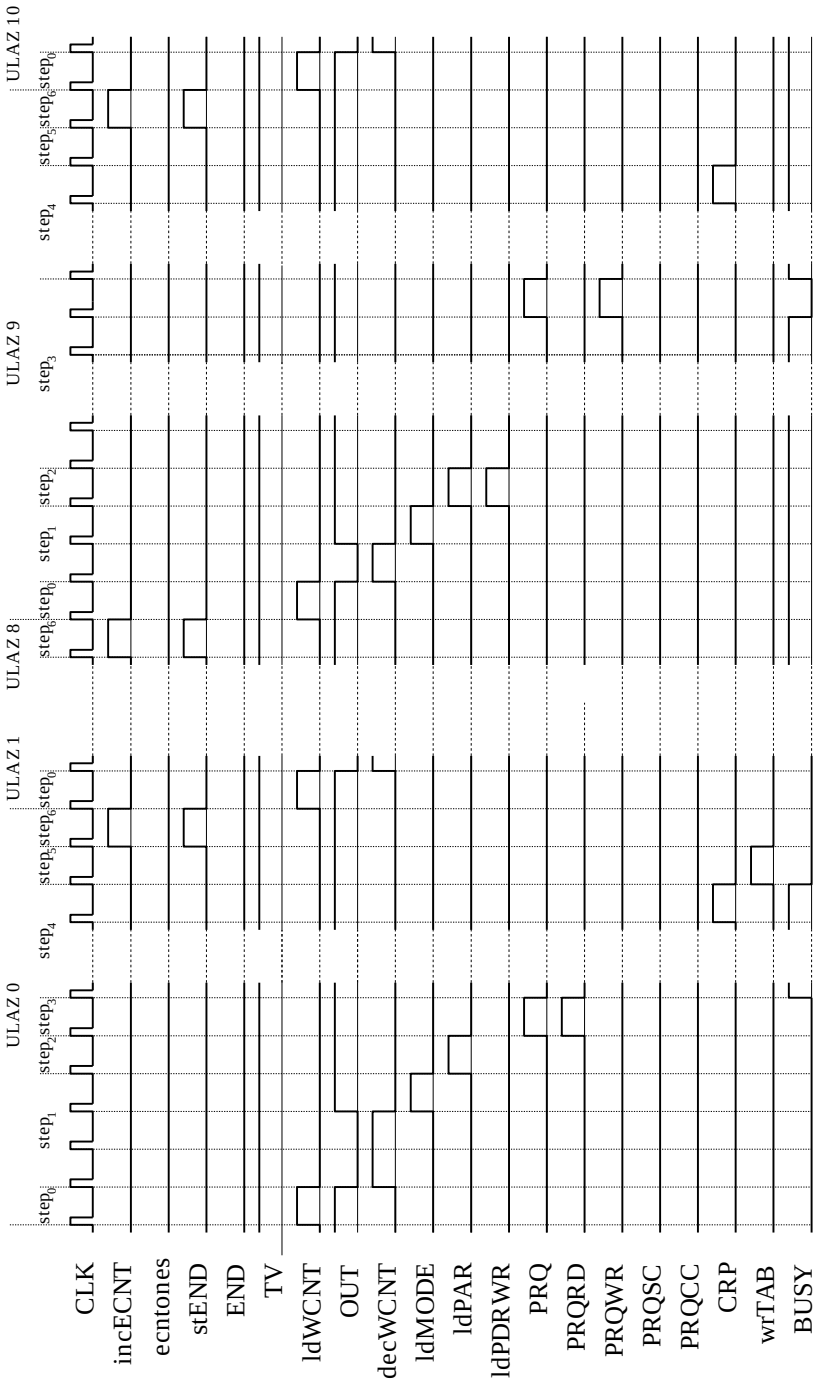
! U korak $step_6$ se dolazi iz koraka $step_5$. U ovom koraku se bezuslovno generišu aktivne vrednost signala **incECNT** i **stEND** bloka *generisanje operacija* trajanja jedna perioda signala takta. Aktivnom vrednošću signala **incECNT** se na signal takta inkrementira sadržaj brojača ECNT i time njegova vrednost podešava na prvi sledeći ulaz memorije TAB bloka *generisanje operacija*. Aktivnom vrednošću signala **stEND** treba da se na signal takta u ovom zadnjem koraku generisanja operacije 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 $step_6$ se ostaje samo jadna perioda signala takta i bezuslovno se prelazi u korak $step_0$.

1.2.2.1.2.3. Vremenski oblici signala

Vremenski oblici signala za operacije čitanja i upisa dati su na slici 42. Oni su veoma slični vremenskim oblicima signala za operacije čitanja i upisa datim na slikama 10 i 12 u odeljku 1.2.1.1.2.3. Jedina razlika je u tome što je na slici 42 u koraku $step_3$ potrebna i neaktivna vrednost signala **BUSY** bloka *keš interfejs* da bi se generisala aktivna vrednost signala **PRQ** bloka *keš interfejs* i prešlo na korak $step_4$. Keš memorija ***KEŠ*** generisanjem aktivne vrednosti signal **CRP** dozvoljava procesoru ***CPU*** da produži sa radom, a aktivnom vrednošću signala **BUSY** sprečava procesor ***CPU*** da u keš memoriji ***KEŠ*** eventualno startuje novu operaciju čitanja ili upisa pre nego što je tekuća operacija čitanja ili upisa završena. Zbog toga mogu da se jave dve situacije. Prva situacija se javlja ukoliko je signal **BUSY** već neaktivan kada se pređe na korak $step_3$. Tada se u koraku $step_3$ ostaje samo jedna perioda signal takta, generiše aktivna vrednost signala **PRQ** i prelazi na korak $step_4$. Na signal takta na koji se prelazi iz koraka $step_3$ na korak $step_4$ generiše se aktivna vrednost signal **BUSY**. Ova situacija je ilustrovana vremenskim oblicima signala za operaciju čitanja za koju je uzeto da se generiše za ulaz 0 memorije TAB bloka *generisanje operacija*. Druga situacija se javlja ukoliko je signal **BUSY** još uvek aktivan kada se pređe na korak $step_3$. Tada se u koraku $step_3$ ostaje sve dok signal **BUSY** ne postane neaktivan. Na signal takta na koji signal **BUSY** postaje neaktivan, signal **PRQ** postaje aktivan. Na sledeći signal takta signal **PRQ** postaje neaktivan i prelazi se na korak $step_4$. Na signal takta na koji se prelazi iz koraka $step_3$ na korak $step_4$

generiše se aktivna vrednost signala **BUSY**. Ova situacija je ilustrovana vremenskim oblicima signala za operaciju upisa za koju je uzeto da se generiše za ulaz 9 memorije TAB.

Vremenski oblici signala koji prikazuju kako se zaustavlja generisanje operacija i po generisanju operacije iz zadnjeg ulaza memorije TAB i kada se u memoriji TAB naiđe na ulaz koji nije važeći su identični sa odgovarajućim vremenskom oblicima signala datim na slikama 11 i 12, pa zato nisu posebno prikazani.



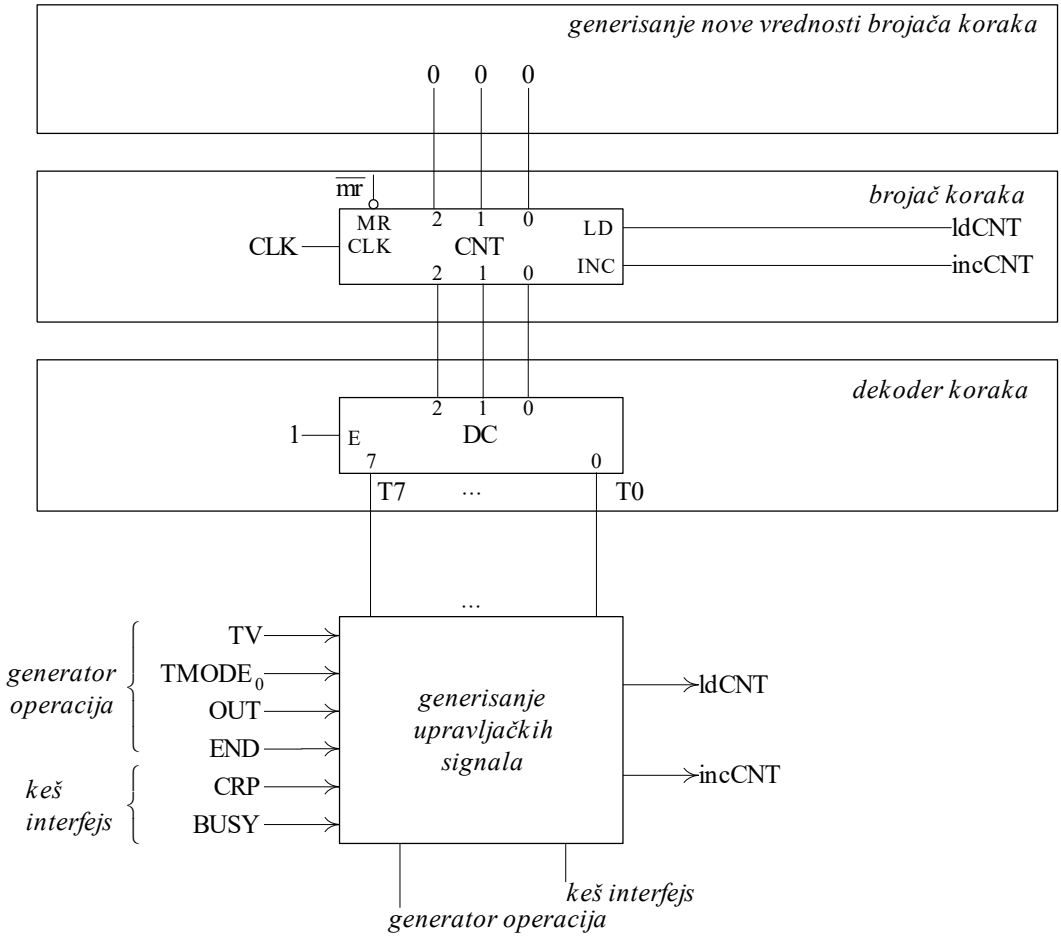
Slika 42 Vremenski oblici signala u procesoru

1.2.2.1.2.4. Struktura upravljačke jedinice

Struktura ***upravljačke jedinice*** ožičene realizacije je prikazana na slici 43. ***Upravljačka jedinica*** 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.



Slika 43 Struktura upravljačke jedinice

1.2.2.1.2.4.1. Blok generisanje nove vrednosti brojača koraka

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 (poglavlje 1.2.2.1.2.2) se utvrđuje da je 0 vrednost koju treba upisati u brojač koraka CNT. Ta vrednost je ožičena na ulazima 0, 1 i 2 brojača CNT.

1.2.2.1.2.4.2. Blok brojač koraka

Blok *brojač koraka* sadrži brojač CNT. Brojač CNT svojom trenutnom vrednošću određuje koji od upravljačkih signala mogu tada da imaju aktivne vrednosti. 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 1.2.2.1.2.2). 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. 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**.

1.2.2.1.2.4.3. Blok dekoder koraka

Blok *dekoder koraka* sadrži dekoder DC. Na ulaze dekodera DC vode se izlazi brojača CNT. Dekodovana stanja brojača CNT pojavljuju se kao signali **T0** do **T7** na izlazima dekodera DC. Svakom koraku iz algoritma generisanja upravljačkih signala (poglavlje 1.2.2.1.2.2) dodeljen je jedan od ovih signala i to koraku $step_0$ signal **T0**, koraku $step_1$ signal **T1**, itd.

1.2.2.1.2.4.4. Blok generisanje upravljačkih signala

Blok *generisanje upravljačkih signala* se sastoji od kombinacionih mreža koje pomoću signala **T0** do **T7** koji dolaze iz bloka *dekoder koraka*, signala logičkih uslova koji dolaze iz blokova ***operacione jedinice*** i saglasno algoritmu generisanja upravljačkih signala (poglavlje 1.2.2.1.2.2) generišu upravljačke signale. Postupak projektovanja kombinacionih mreža je identičan sa postupkom koji je objašnjen u poglavlju 1.2.1.1.2.4.4.

Blok *generisanje upravljačkih signala* generiše dve grupe upravljačkih signala i to:
- upravljačke signale operacione jedinice i
- upravljačke signale upravljačke jedinice.

Generisanje upravljačkih signala operacione jedinice i upravljačkih signala upravljačke jedinice se daje u daljem tekstu.

1.2.2.1.2.4.4.1. Upravljački signali operacione jedinice

Upravljački signali operacione jedinice podeljeni su u grupe koje odgovaraju blokovima ***operacione jedinice*** i to:
- blok *keš interfejs* i
- blok *generator operacija*.

1.2.2.1.2.4.4.1.1. Blok keš interfejs

Upravljački signali koji odgovaraju ovom bloku ***operacione jedinice*** su:
- **ldPDRWR**—signal paralelnog upisa u registar PDRWR,
- **ldMODE**—signal paralelnog upisa u registar MODE,
- **ldPAR**—signal paralelnog upisa u registar PAR i
- **PRQ**—signal startovanja jedne od dve operacije keš memorije ***KEŠ***.

Ovi upravljački signali se generišu na sledeći način:

- **ldPDRWR** = **T2** · **TMODE$_0$**,
- **ldMODE** = **T2** · **OUT**,
- **ldPAR** = **T2** i
- **PRQ** = **T3** · **BUSY**.

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

- **TMODE$_0$** — blok *generisanje operacija*,
- **OUT** — blok *generisanje operacija* i
- **BUSY** — blok *keš interfejs*.

1.2.2.1.2.4.4.1.2. Blok generisanje operacija

Upravljački signali koji odgovaraju ovom bloku ***operacione jedinice*** su:

- **ldWCNT**— signal paralelnog upisa u brojač WCNT,
- **decWCNT**— signal dekrementiranja sadržaja brojača WCNT,
- **incECNT**— signal inkrementiranja sadržaja brojača ECNT,
- **wrTAB** — signal upisa sadržaja u memoriju TAB i
- **stEND** — signal postavljanja flip-flopa END.

Ovi upravljački signali se generišu na sledeći način:

- **ldWCNT** = **T0** · **TV** · $\overline{\textbf{END}}$,
- **decWCNT** = **T1** · $\overline{\textbf{OUT}}$,
- **incECNT** = **T6**,
- **wrTAB** = **T5** · **TMODE$_0$** i
- **stEND** = **T6**.

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

- **TV** — blok generisanje operacija,
- **END** — blok generisanje operacija,
- **OUT** — blok generisanje operacija i
- **TMODE$_0$** — blok generisanje operacija.

1.2.2.1.2.4.4.2. Upravljački signali upravljačke jedinice

Upravljački signali upravljačke jedinice su:

- **ldCNT** — signal paralelnog upisa u brojač CNT i
- **incCNT** — signal inkrementiranja sadržaja brojača CNT.

Ovi upravljački signali se generišu na sledeći način:

- **ldCNT** = **T6** i
- **incCNT** = **T0** · **TV** · $\overline{\textbf{END}}$ + **T1** · **OUT** + **T2** + **T3** · $\overline{\textbf{BUSY}}$ + **T4** · **CRP** + **T5**.

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

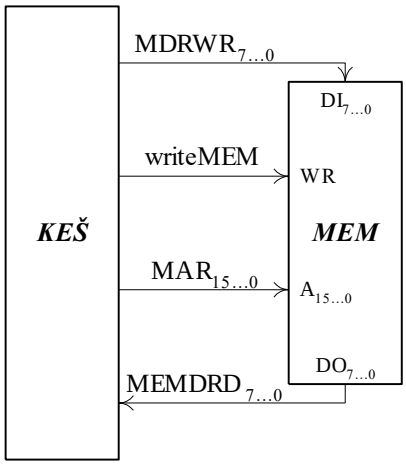
- **TV** — blok *generisanje operacija*,
- **END** — blok *generisanje operacija*,
- **OUT** — blok *generisanje operacija*,
- **BUSY** — blok *keš interfejs* i

- **CRP** — blok *keš interfejs*.

1.2.2.2. MEMORIJA MEM

Memoriji ***MEM*** se obraća keš memorija ***KEŠ*** ili kada u okviru operaciju upisa treba realizovati upise u memorije ***MEM*** ili kada u okviru operacije čitanja u keš memoriji ***KEŠ*** nije otkrivena saglasnost pa treba prilikom prebacivanja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** realizovati čitanja iz memorije ***MEM*** (slika 44).

Kod čitanja iz memorije ***MEM*** keš memorija ***KEŠ*** šalje 16-bitnu adresu po linijama $\mathbf{MAR_{15...0}}$ i drži neaktivnu vrednost signala **writeMEM**. Očitani 8-bitni podatak se vraća po linijama $\mathbf{MEMDRD_{7...0}}$. Kod upisa u memoriju ***MEM*** keš memorija ***KEŠ*** šalje 16-bitnu adresu po linijama $\mathbf{MAR_{15...0}}$, 8-bitni podatak po linijama $\mathbf{MDRWR_{7...0}}$ i drži aktivnu vrednost signala **writeMEM**. Uzeto je da je vreme pristupa memoriji ***MEM*** jednako deset perioda signala takta.



Slika 44 Memorija MEM

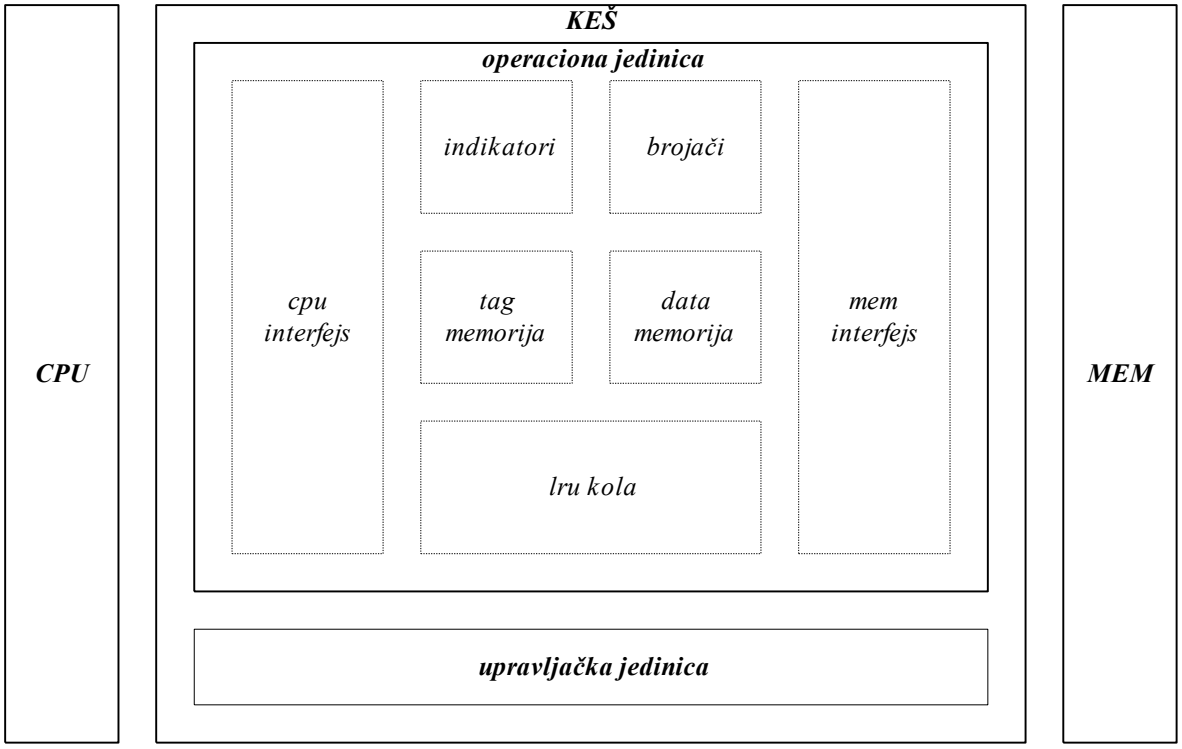
1.2.2.3. KEŠ MEMORIJA KEŠ

Keš memorija ***KEŠ*** (slika 45) 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 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 prema algoritmu generisanja upravljačkih signala keš memorije ***KEŠ*** i signala logičkih uslova.

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



Slika 45 Keš memorija KEŠ

1.2.2.3.1. Operaciona jedinica

Operaciona jedinica (slika 45) se sastoji iz sledećih blokova:

- blok *cpu interfejs,*
- blok *indikatori,*
- blok *brojači,*
- blok *tag memorija,*
- blok *data memorija,*
- blok *lru kola* i
- blok *mem interfejs*.

Blok *cpu interfejs* služi za razmenu adresa, podataka i upravljačkih signala između keš memorije ***KEŠ*** i procesora ***CPU***.

Blok *indikatori* služi za vođenje evidencije za svaki od 8 ulaza keš memorije ***KEŠ*** o tome da li je ulaz važeći.

Blok *brojači* služi za generisanje adresa bajtova unutar bloka, za prebrojavanje broja prenetih bajtova iz memorije ***MEM*** u keš memoriju ***KEŠ*** i odbrojavanje onoliko perioda signala takta koliko je vreme pristupa memoriji ***MEM***.

Blok *tag memorija* služi za vođenje evidencije za svaki od 8 ulaza keš memorije ***KEŠ*** o tome kom bloku memorije ***MEM*** pripada blok podataka koji se nalazi u datom ulazu keš memorije ***KEŠ***.

Blok *data memorija* služi za smeštanje 8 blokova podataka.

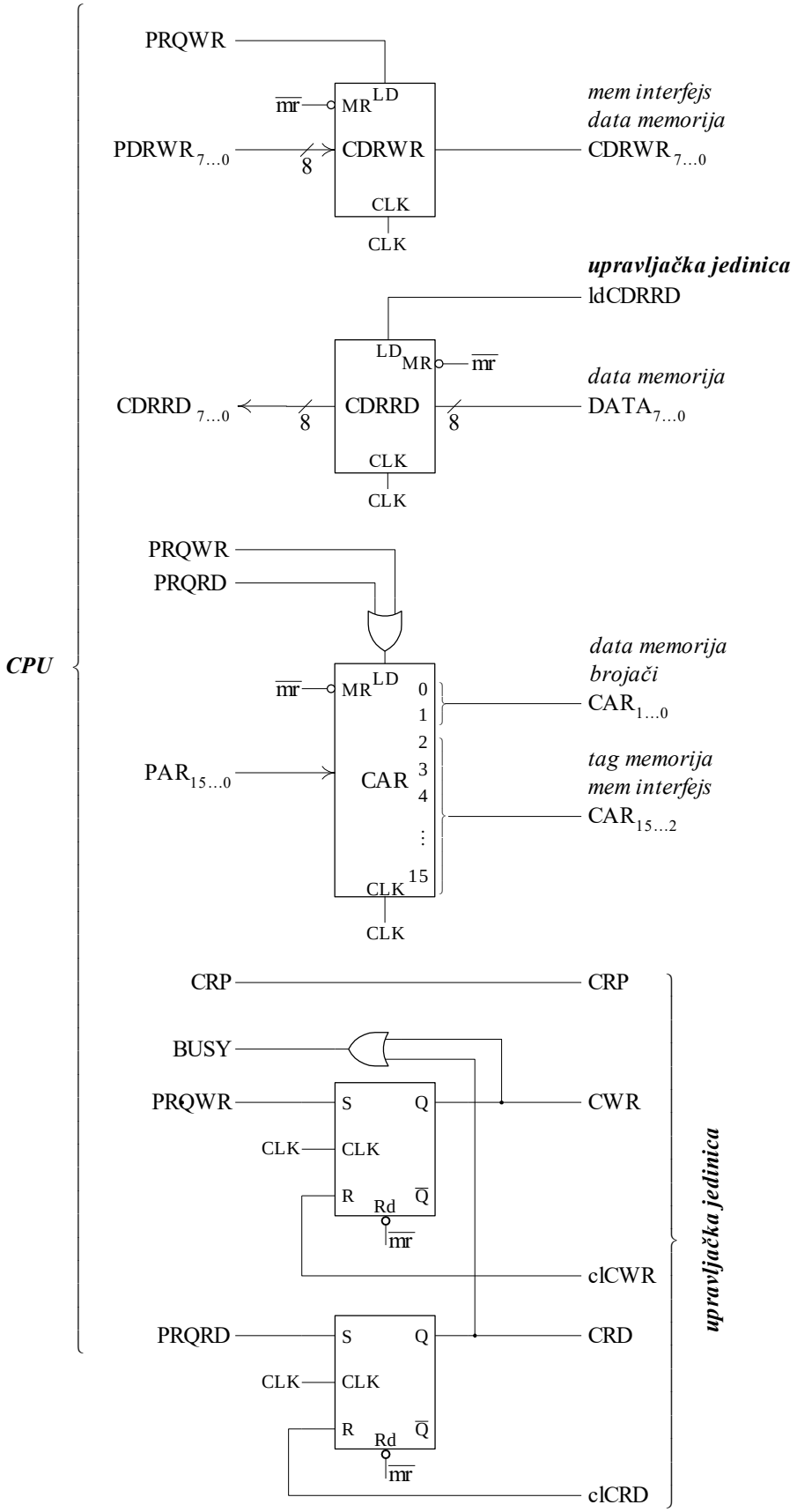
Blok *lru kola* služi za implementaciju LRU algoritma zamene blokova keš memorije ***KEŠ***.

Blok *mem interfejs* služi za razmenu adresa, podataka i upravljačkih signala između keš memorije ***KEŠ*** i memorije ***MEM***.

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

1.2.2.3.1.1. Blok cpu interfejs

Blok *cpu interfejs* (slika 46) sadrži registre CAR, CDRWR i CDRRD i flip-flopove CRD i CWR.



Slika 46 Blok cpu interfejs

Registar CAR (Cache Address Register) služi za čuvanje ili 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. Na ulaze registra CAR dovodi se 16-bitna adresa po linijama $\mathbf{PAR_{15...0}}$ i upisuje u registar CAR pri pojavi signala takta ukoliko je aktivan jedan od signala **PRQRD** ili **PRQWR** .

Razredi $\mathbf{CAR_{15...2}}$ se vode u:

- blok *tag memorija* radi utvrđivanja saglasnosti, a kod dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** i radi njihovog upisivanja u memorijski modul TAG i
- blok *mem interfejs* gde se koriste u formiranju adrese memorije ***MEM*** kod dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** ili prilikom upisa bajta podatka u memoriju ***MEM***.

Razredi $\mathbf{CAR_{1...0}}$ se vode u:

- u blok *data memorija* gde se koriste za adresiranje bajta u bloku kome treba pristupiti i
- u blok *brojači* gde se koriste kao adresa bajta memorije ***MEM*** u koji se vrši upis u slučaju operacije upisa ili kao adresa bajta memorije ***MEM*** od koje se u slučaju operacije čitanja po otkrivanju nesaglasnosti kreće sa čitanjem bajtova bloka koji se dovlače iz memorije ***MEM*** u keš memoriju ***KEŠ***.

Registar CDRWR (Cache Data Register for WRite) služi za čuvanje podatka koji treba upisati u slučaju operacije upisa. Na ulaze registra CDRWR dovodi se po linijama $\mathbf{PDRWR_{7...0}}$ 8-bitni podatak i upisuje u registar CDRWR pri pojavi signala takta ukoliko je aktivan signal **PRQWR**. Sadržaj registra CDRWR se preko bloka *mem interfejs* uvek upisuje u memoriju ***MEM***, dok se u memorijski modul DATA bloka *data memorija* upisuje u slučaju operacije upisa ukoliko je otkrivena saglasnost.

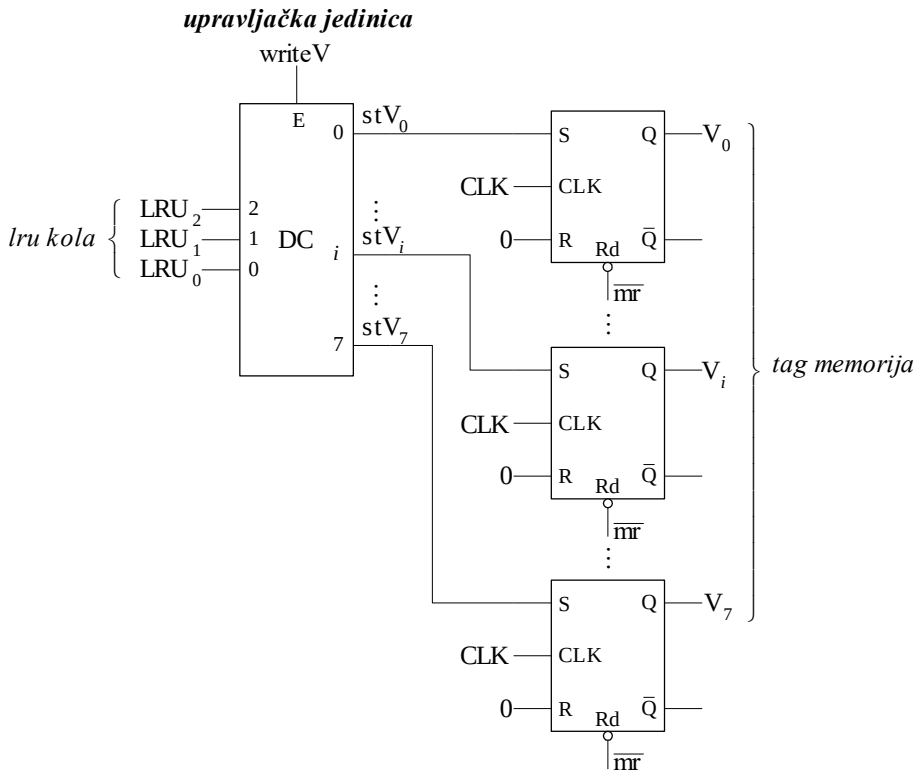
Registar CDRRD (Cache Data Register for ReaD) služi za čuvanje očitanog podatka u slučaju operacije čitanja. Na ulaze registra CDRRD dovodi se po linijama $\mathbf{DATA_{7...0}}$ 8-bitni podatak iz memorijskoj modula DATA bloka *data memorija* i upisuje u registar CDRRD pri pojavi signala takta ukoliko je aktivan signal **ldCDRRD**.

Aktivnom vrednošću signala **PRQRD** ili **PRQWR** trajanja jedna perioda signala takta procesor ***CPU*** startuje u keš memoriji ***KEŠ*** operaciju čitanja ili upisa. Aktivnom vrednošću signala **CRP** trajanja jedna perioda signala takta keš memorija ***KEŠ*** signalizira procesoru ***CPU*** da može da nastavi sa radom bez obzira na to da li je odgovarajuća operacija kompletirana u keš memorije ***KEŠ***.

Flip-flopovi CRD (Cache ReaD) i CWR (Cache WRite) služe da se u keš memoriji ***KEŠ*** upamti da su u toku operacije čitanja ili upisa. Flip-flop CRD se postavlja na aktivnu vrednost na signal takta pri pojavi aktivne vrednosti signala **PRQRD** trajanja jednu periodu signala takta. Aktivna vrednost ostaje sve dok je operacija čitanja u toku. Flip-flop CRD se postavlja na neaktivnu vrednost na kraju operacije čitanja na signal takta pri aktivnoj vrednosti signala **clCRD** trajanja jedna perioda signala takta. Flip-flop CWR se postavlja na aktivnu vrednost prilikom startovanja operacije upisa pri pojavi aktivne vrednosti signala **PRQWR**, a na neaktivnu vrednost pri aktivnoj vrednosti signala **clCWR**.

1.2.2.3.1.1. Indikatori

Blok *indikatori* (slika 47) sadrže flip-flopove V_0 do V_7 i dekoder DC.



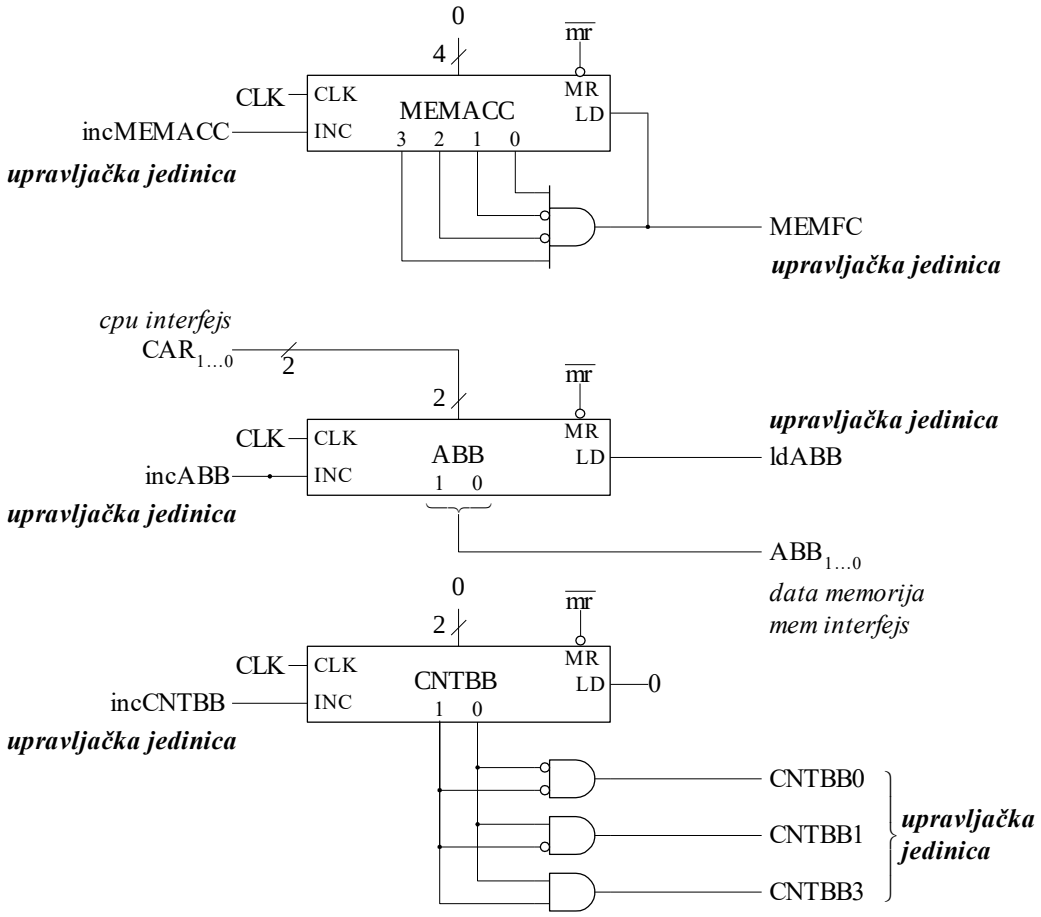
Slika 47 Blok *indikatori*

Flip-flopovi V_0 do V_7 služe za čuvanje indikatora V (Valid). Njima se za svaki ulaz keš memorije ***KEŠ*** vodi evidencija da li je važeći ili ne. Dekoder DC služi za formiranje signala **stV₀** do **stV₇**. U slučaju operacija čitanja se prilikom dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** generiše aktivna vrednost signala **writeV**. Tada se u zavisnosti od vrednosti signala **LRU₂...₀** generiše aktivna vrednost jednog od signala **stV₀** do **stV₇** i time određuje jedan od flip-flopova V_0 do V_7 u koji se na signal takta upisuje aktivna vrednost.

1.2.2.3.1.2. Blok brojači

Blok *brojači* (slika 48) sadrži brojač bajtova bloka CNTBB (CouNTer of Block Bytes), brojač adresa bajtova bloka ABB (Addresses of Block Bytes) i brojač vremena pristupa operativnoj memoriji MEMACC (MEMory ACCess time).

Brojač bajtova bloka CNTBB se koristi za odbrojavanje broja prenetih bajtova prilikom prenosa bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ***. Brojač CNTBB je realizovan kao inkrementirajući brojač po modulu četiri, jer je četiri veličina bloka u bajtovima. Ovaj brojač uvek kreće sa brojanjem od vrednosti nula, a njegova vrednost tri je indikaciju da se prenosi zadnji bajt bloka i da prenos treba prekinuti. Inkrementanje sadržaja brojača CNTBB se realizuje pri pojavi signala takta i aktivnoj vrednosti signala **incCNTBB**. Signali **CNTBB3**, **CNTBB1** i **CNTBB0** služe kao indikacija da su sadržaji brojača CNTBB 3, 1 i 0, respektivno.



Slika 48 Blok brojači

Brojač adresa bajtova bloka ABB se koristi za pamćenje adrese bajta u bloku prilikom čitanja iz keš memorije ***KEŠ***, upisa u keš memoriju ***KEŠ*** i upisa u memoriju ***MEM*** i za generisanje adresa bajtova u bloku prilikom dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ***. Prilikom svake operacije čitanja i upisa sadržaj sa linija $\mathbf{CAR_{1...0}}$, koji predstavlja adresu bajta u bloku generisane adrese, se na signala takta i aktivnoj vrednosti signala **ldABB** upisuje u brojač ABB. Inkrementanje sadržaja brojača ABB se realizuje pri pojavi signala takta i aktivnoj vrednosti signala **incABB**. Brojač ABB se koristi na različite načine u zavisnosti od toga da li ima ili nema saglasnosti i da li je reč o operaciji čitanja ili upisa. Brojač ABB se koristi u sledećim blokovima:

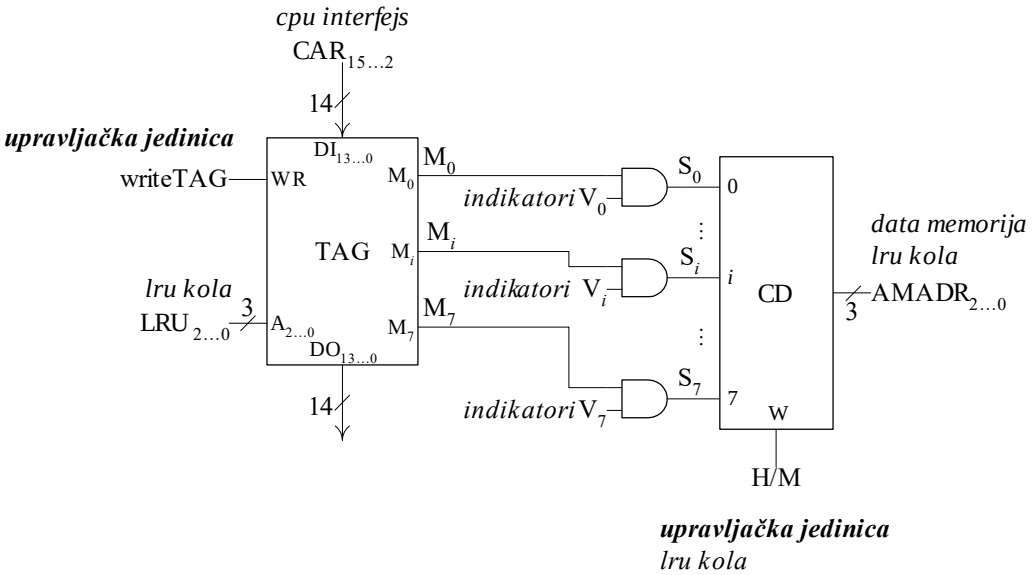
1. blok *data memorija*. U slučaju da je za operaciju čitanja otkriveno da ima saglasnosti, sadržaj brojača ABB se koristi kao adresa bajta u bloku ulaza keš memorije ***KEŠ*** u kome je otkrivena saglasnost da se na njoj realizuje čitanje bajta podatka. U slučaju da je za operaciju čitanja otkriveno da nema saglasnosti, sadržaj brojača ABB se koristi kao adresa bajta u bloku počev od koje se upisuju četiri bajta podatka bloka koji se dovlači iz memorije ***MEM*** u ulaz keš memorije ***KEŠ*** koji je odabran za zamenu. U slučaju da je za operaciju upisa otkriveno da ima saglasnosti, sadržaj brojača ABB se koristi kao adresa bajta u bloku ulaza keš memorije ***KEŠ*** u kome je otkrivena saglasnost da se na njoj realizuje upis bajta podatka. U slučaju da je za operaciju upisa otkriveno da nema saglasnosti ne pristupa se keš memoriji ***KEŠ***, pa se u ovom slučaju sadržaj brojača ABB ne koristi.
2. blok *mem interfejs*. U slučaju da je za operaciju čitanja otkriveno da ima saglasnosti ne pristupa se memoriji ***MEM***, pa se u ovom slučaju sadržaj brojača ABB ne koristi. U slučaju da je za operaciju čitanja otkriveno da nema saglasnosti sadržaj brojača ABB se

koristi kao adresa bajta u bloku počev od koje se čitaju iz memorije ***MEM*** četiri bajta bloka podataka koji se dovlači iz memorije ***MEM*** u ulaz keš memorije ***KEŠ*** koji je odabran za zamenu. U slučaju operacije upisa, bez obzira na to da li ima ili nema saglasnosti, sadržaj brojača ABB se koristi kao adresa bajta u bloku memorije ***MEM*** na kojoj se realizuje upis bajta podatka.

Brojač vremena pristupa operativnoj memoriji MEMACC se koristi da odbroji deset signala takta kod obraćanja keš memorije ***KEŠ*** memoriji ***MEM*** radi čitanja ili upisa. Usvojeno je da deset signala takta vreme pristupa memoriji ***MEM***. Brojač MEMACC je realizovan kao inkrementirajući brojač po modulu deset koji uvek kreće sa brojanjem od vrednosti nula. Inkrementanje sadržaja brojača MEMACC se realizuje pri pojavi signala takta i aktivnoj vrednosti signala **incMEMACC**. Signal **MEMFC** (MEMory Function Completed) postaje aktivan kada brojač MEMACC pređe u stanje devet. Na deseti signal takta brojač MEMACC se vraća u stanje nula, a signal **MEMFC** postaje neaktivan. Signal **MEMFC** se koristi kao signal logičkog uslova da je pristup memoriji ***MEM*** završen.

1.2.2.3.1.3. Blok tag memorija

Blok *tag memorija* (slika 49) sadrži asocijativni memorijski modul TAG, koder CD i 8 logičkih I kola.



Slika 49 Blok tag memorija

Memorijski modul TAG čuva za svaki od 8 ulaza keš memorije ***KEŠ*** broj bloka memorije ***MEM*** koji se trenutno nalazi u datom ulazu keš memorije ***KEŠ***. Kapacitet memorijskog modula TAG je 8 reči širine 14 bitova. Memorijski modul TAG ima sledeće ulaze i izlaze:

- adresne linije $A_{2...0}$,
- ulazne linije podataka $DI_{13...0}$,
- izlazne linije podataka $DO_{13...0}$,
- izlazne liije podudaranja $M_{7...0}$ i
- upravljačku liniju WR.

Upis u memorijski modul TAG se realizuje aktivnom vrednošću signala **writeTAG**, a čitanje njegovom neaktivnom vrednošću. Pored toga u memorijskom modulu TAG se realizuje i upoređivanje sadržaja svake od osam lokacija sa sadržajima na ulaznim linijama podataka $DI_{13...0}$. Na osnovu toga se formiraju signali podudaranja **M** (Match) za svaki od

osam ulaza memorijskog modula TAG. Ovi signali su označeni sa $\mathbf{M_7}$ do $\mathbf{M_0}$ i pojavljuju se na izlaznim linijama podudaranja $\mathbf{M_{7...0}}$. Memorijskom modulu TAG se pristupa u sledećim slučajevima:

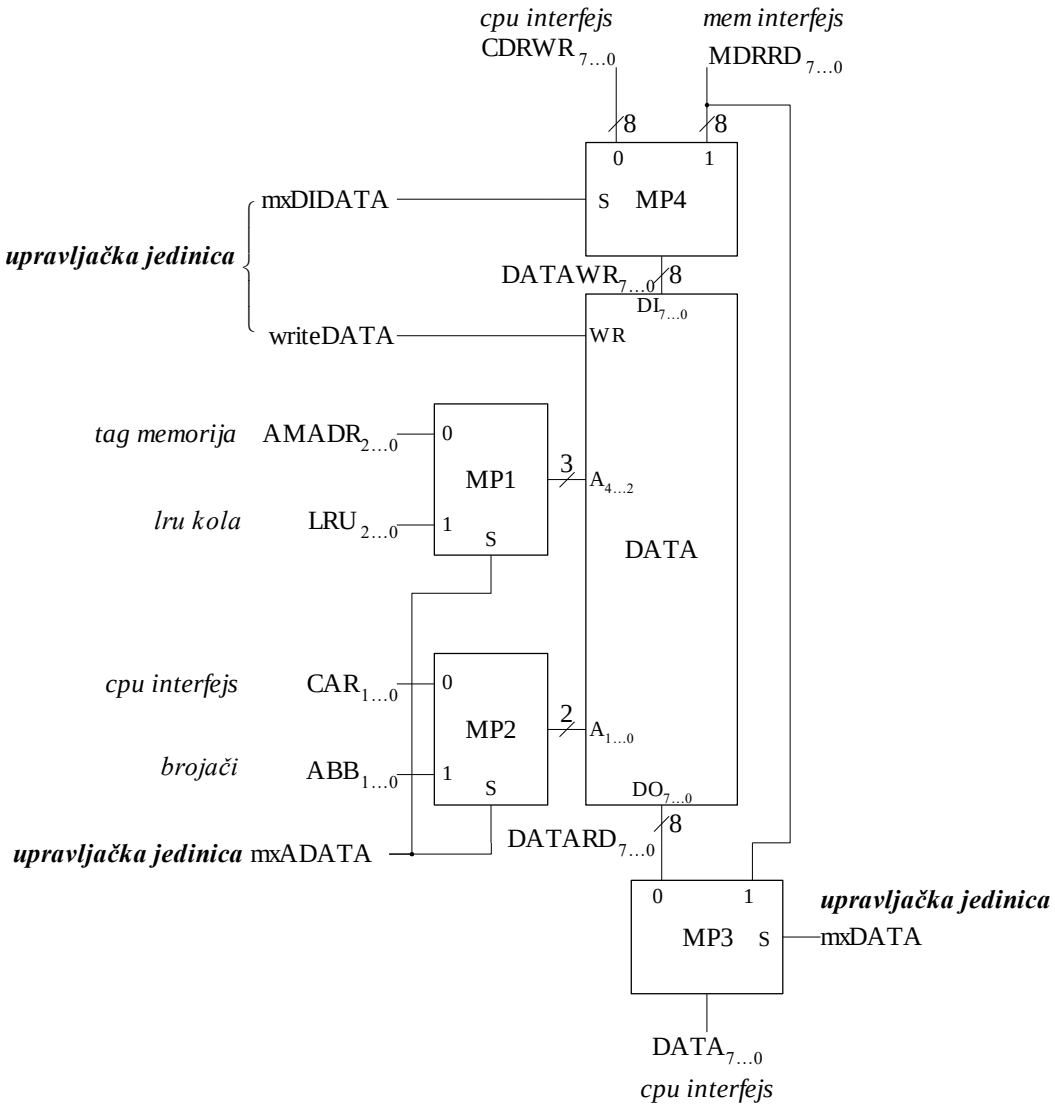
1. *Kod operacija čitanja i upisa da bi se utvrdilo da li ima saglasnosti*. U ovom slučaju na ulazne linije podataka $DI_{13...0}$ se vode signali $\mathbf{CAR_{15...2}}$. Ovi signali predstavljaju broj bloka memorije ***MEM*** kome pripada generisana adresa. Ukoliko se dati blok nalazi u nekom od osam ulaza keš memorije ***KEŠ*** doći će do podudaranja sadržaja datog ulaza memorijskog modula TAG i sadržaja na ulaznim linijama podataka $DI_{13...0}$ i odgovarajući signal **M** na izlaznim linijama podudaranja $\mathbf{M_{7...0}}$ će postati aktivan. U suprotnom slučaju svi signali $\mathbf{M_7}$ do $\mathbf{M_0}$ će biti neaktivni.
2. *Kod operacije čitanja pri upisu u modul DATA bajtova bloka podataka koji se dovlače iz memorije **MEM** u keš memoriju **KEŠ** ako nema saglasnosti*. Signali na ulaznim linijama podataka $DI_{13...0}$ imaju isto značenje i formiraju se na isti način kao u slučaju 1. Sadržaj sa linija $DI_{13...0}$ se upisuje u isti ulaz modula TAG kao što je i ulaz modula DATA u koji se iz memorije ***MEM*** dovlači blok podataka. Taj ulaz je određen vrednošću signala $\mathbf{LRU_{2...0}}$ koji se vode na adresne linije $A_{2...0}$ modula TAG. Upis se realizuje na aktivnu vrednost signala **writeTAG**.

Logička I kola služe za formiranje signal saglasnosti $\mathbf{S_7}$ do $\mathbf{S_0}$ za svaki od osam ulaza keš memorije ***KEŠ***. Ovi signali se formiraju prema relaciji $\mathbf{S}_i = \mathbf{M}_i * \mathbf{V}_i$, 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 ($\mathbf{V}_i = 1$).

Koder CD služi za formiranje signala **H/M** i signala binarne vrednosti ulaza keš memorije ***KEŠ*** u kome je otkrivena saglasnost $\mathbf{AMADR_{2...0}}$. Signal **H/M** je aktivan ukoliko je jedan od signala $\mathbf{S_7}$ do $\mathbf{S_0}$ aktivan. U suprotnom slučaju, signal **H/M** je neaktivan. Signali $\mathbf{AMADR_{2...0}}$ se formiraju na osnovu toga koji od signala $\mathbf{S_7}$ do $\mathbf{S_0}$ ima aktivnu vrednost.

1.2.2.3.1.4. Blok data memorija

Blok *data memorija* (slika 50) sadrži memorijski modul DATA i multipleksere MP1, MP2, MP3 i MP4.



Slika 50 Blok data memorija

Memorijski modul DATA čuva sadržaje 8 blokova podataka koji se trenutno nalaze u keš memoriji ***KEŠ***. Ovaj modul ima 32 lokacije širine jedan bajt organizovanih u 8 blokova po 4 bajta. Stoga je adresa modula DATA širine 5 bita, gde niža 2 bita definišu bajt unutar bloka, a viša 3 bita određuju broj bloka keš memorije.

Memirijski modul DATA ima sledeće ulaze i izlaze:

- adresne linije $A_{4...0}$,
- ulazne linije podataka $DI_{7...0}$,
- izlazne linije podataka $DO_{7...0}$ i
- upravljačku liniju WR.

Upis u memorijski modul DATA se realizuje aktivnom vrednošću signala **writeDATA**, a čitanje njegovom neaktivnom vrednošću.

Memorijskom modulu DATA prisupa se u sledećim slučajevima:

1. *Kod operacije čitanja pri čitanju iz keš memorije **KEŠ** bajta podatka ako je otkrivena saglasnost*. U ovom slučaju signali na adresnim linijama $A_{4...2}$ i $A_{1...0}$ memorijskog modula DATA se formiraju od signala $\mathbf{AMADR_{2...0}}$ i $\mathbf{CAR_{1...0}}$ koji se kroz multipleksere MP1 i MP2 propuštaju neaktivnom vrednošću upravljačkog signala **mxADATA**. Očitani podatak se pojavljuje kao signali $\mathbf{DATARD_{7...0}}$ na izlaznim

linijama podataka $DO_{7...0}$ memorijskog modula DATA, propušta kroz multiplekser MP3 neaktivnom vrednošću upravljačkog signala **mxDATA** i pojavljuje na izlazima multipleksera MP3 kao signali $\mathbf{DATA_{7...0}}$.

2. *Kod operacije upisa pri upisu u keš memoriju **KEŠ** bajta podatka ako je otkrivena saglasnost*. Adresa se formira kao u slučaju 1. Podatak za upis u memorijski modul DATA se formira od signala $\mathbf{CDRWR_{7...0}}$ koji se propuštaju kroz multiplekser MP4 neaktivnom vrednošću upravljačkog signala **mxDIDATA** i pojavljuju kao signali $\mathbf{DATAWR_{7...0}}$ na ulaznim linijama podataka $DI_{7...0}$ memorijskog modula DATA.
3. *Kod operacije čitanja pri upisu u keš memoriju **KEŠ** bajtova podataka bloka koji se dovlače iz memorije **MEM** u keš memoriju **KEŠ** ako nema saglasnosti*. U ovom slučaju signali na adresnim linijama $A_{4...2}$ i $A_{1...0}$ memorijskog modula DATA se formiraju od signala $\mathbf{LRU_{2...0}}$ i $\mathbf{ABB_{1...0}}$ koji se kroz multipleksere MP1 i MP2 propuštaju aktivnom vrednošću upravljačkog signala **mxADATA**. Podatak za upis u memorijski modul DATA se formira od signala $\mathbf{MDRRD_{7...0}}$ koji se propuštaju kroz multiplekser MP4 aktivnom vrednošću upravljačkog signala **mxDIDATA** i pojavljuju kao signali $\mathbf{DATAWR_{7...0}}$ na ulaznim linijama podataka $DI_{7...0}$ memorijskog modula DATA.

Multiplekser MP1 služi za formiranje signala na adresnim linijama $A_{4...2}$ memorijskog modula DATA. Kroz multiplekser MP1 se u slučajevima 1 i 2 neaktivnom vrednošću upravljačkog signala **mxADATA** propuštaju signali $\mathbf{AMADR_{2...0}}$, a u slučaju 3 se aktivnom vrednošću upravljačkog signala **mxADATA** propuštaju signali $\mathbf{LRU_{2...0}}$.

Multiplekser MP2 služi za formiranje signala na adresnim linijama $A_{1...0}$ memorijskog modula DATA. Kroz multiplekser MP2 se u slučajevima 1 i 2 neaktivnom vrednošću upravljačkog signala **mxADATA** propuštaju signali $\mathbf{CAR_{1...0}}$, a u slučaju 3 se aktivnom vrednošću upravljačkog signala **mxADATA** propuštaju signali $\mathbf{ABB_{1...0}}$.

Multiplekser MP3 služi za formiranje signala na ulaznim linijama podataka $DI_{7...0}$ memorijskog modula DATA. Kroz multiplekser MP3 se u slučaju 2 neaktivnom vrednošću upravljačkog signala **mxDIDATA** propuštaju signali $\mathbf{CDRWR_{7...0}}$, a u slučaju 3 se aktivnom vrednošću upravljačkog signala **mxDIDATA** propuštaju signali $\mathbf{MDRRD_{7...0}}$.

Multiplekser MP4 služe za formiranje signala $\mathbf{DATA_{7...0}}$ očitanog bajta podatka. Kroz multiplekser MP3 se u slučaju 1 neaktivnom vrednošću upravljačkog signala **mxDATA** propuštaju signali $\mathbf{DATA_{7...0}}$ koji predstavljaju očitani podatak iz keš memorije ***KEŠ***, a u slučaju 3 se aktivnom vrednošću upravljačkog signala **mxDATA** propuštaju signali $\mathbf{MDRRD_{7...0}}$ koji predstavljaju očitani podatak iz memorije ***MEM***. Ovim se u slučaju 3 realizuje ne samo upis očitanog bajta podatka iz memorije ***MEM*** u memorijski modul DATA keš memorije ***KEŠ***, već i njegovo prosleđivanje u procesor ***CPU***.

1.2.2.3.1.5. Blok lru kola

Blok *lru kola* se koristi za realizaciju LRU algoritma zamene. U ovom odeljku se daju prikaz realizovanog LRU algoritma zamene i strukturna šema bloka *lru kola*.

1.2.2.3.1.5.1. Algoritam zamene

Realizovani algoritam zamene podrazumeva korišćenje posebnog brojača po modulu broj ulaza keš memorije ***KEŠ*** za svaki ulaz keš memorije ***KEŠ***. Ovi brojači se u daljem tekstu nazivaju LRU brojači. U datom slučaju se koristi 8 LRU brojača po modulu 8. LRU brojači se ažuriraju na takav način da je pri popunjenoj keš memoriji ***KEŠ*** u njima 8 različitih vrednosti od 0 do 7. LRU brojač sa sadržajem 7 pripada ulazu keš memorije ***KEŠ*** kome je najmanje skoro pristupano, LRU brojač sa sadržajem 6 pripada ulazu keš memorije ***KEŠ*** kome je nešto

skorije pristupano i tako redom do LRU brojača sa sadržajem 0 koji pripada ulazu keš memorije ***KEŠ*** kome se pristupilo kod zadnjeg pristupa keš memoriji ***KEŠ***.

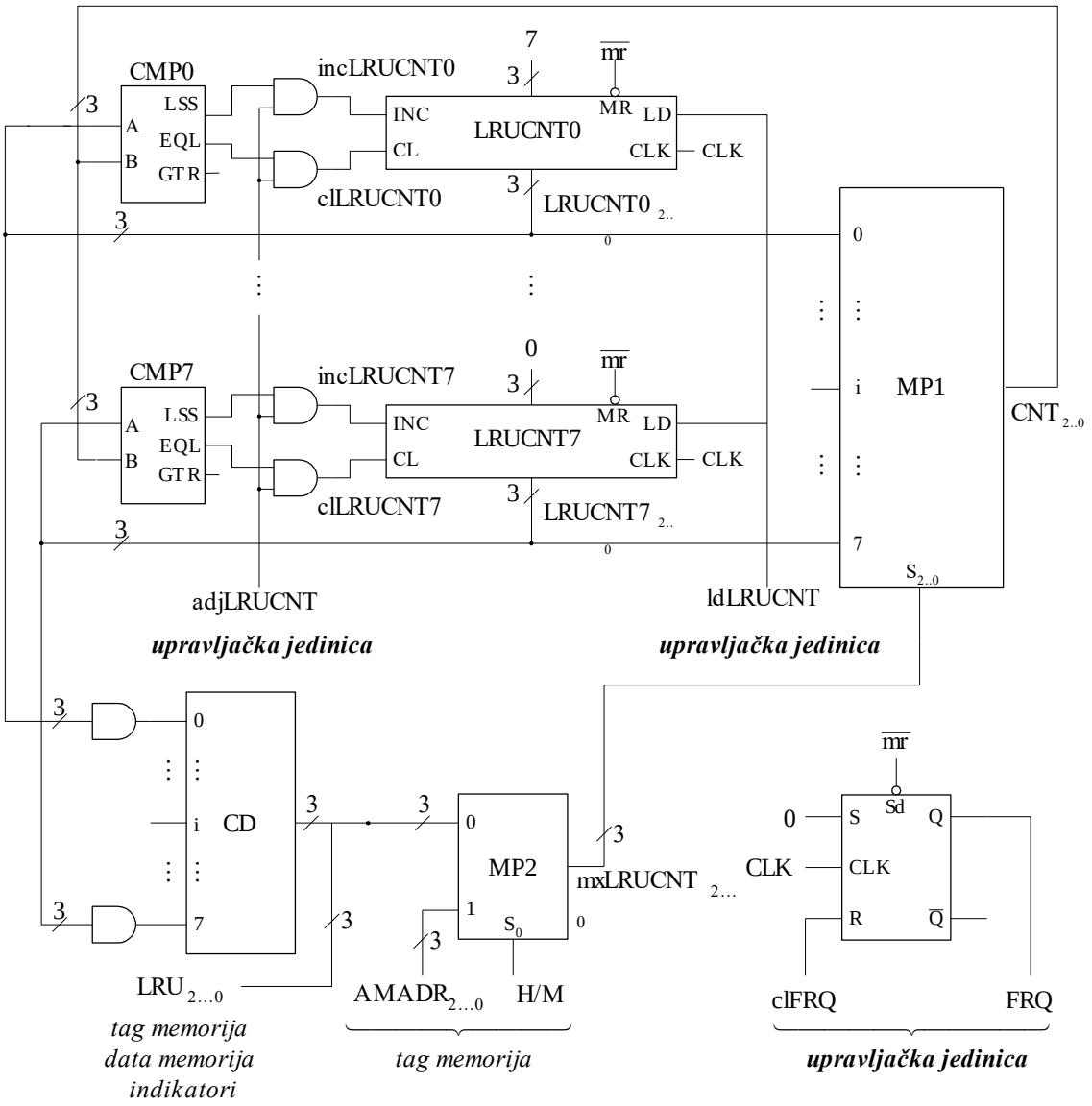
Kod svakog pristupa keš memoriji ***KEŠ*** kod koga se u nekom ulazu keš memorije ***KEŠ*** otkrije saglasnost, pored čitanja iz keš memorije ***KEŠ*** ili upisa u keš memoriju ***KEŠ***, vrši se i ažuriranje LRU brojača. Sadržaj LRU brojača ulaza u kome je otkrivena saglasnost se poredi sa sadržajima LRU brojača preostalih 7 ulaza keš memorije ***KEŠ***. LRU brojači čiji su sadržaji manji od sadržaja LRU brojača ulaza keš memorije ***KEŠ*** u kome je otkrivena saglasnost se inkrementiraju, sadržaji LRU brojači čiji su sadržaji veći od sadržaja LRU brojača ulaza keš memorije ***KEŠ*** u kome je otkrivena saglasnost se ne menjaju i sadržaj LRU brojača ulaza keš memorije ***KEŠ*** u kome je otkrivena saglasnost se postavlja na vrednost nula.

Kod svakog pristupa keš memoriji ***KEŠ*** kod koga se ni u jednom od ulaza keš memorije ***KEŠ*** ne otkrije saglasnost, ulaz keš memorije ***KEŠ*** čiji LRU brojač ima vrednost 7 se bira za zamenu. U dati ulaz keš memorije ***KEŠ*** se iz memorije ***MEM*** dovlači novi blok podataka, u njegov LRU brojač upisuje vrednost 0, a sadržaji LRU brojača preostalih 7 ulaza keš memorije ***KEŠ*** inkrementiraju.

Isti postupak se koristi i kod početnog popunjavanja ulaza keš memorije ***KEŠ***. Ukoliko se želi da se to čini redom od ulaz 0, zatim ulaza 1 i tako redom do ulaza 7 kao zadnjeg ulaza keš memorije ***KEŠ*** koji se popunjava, onda se na početku kada je keš memorija ***KEŠ*** prazna i svi njeni ulazi nevažeći, u LRU brojače ulaza 0, 1 i tako redom do LRU brojača ulaza 7 upisuju vrednosti 7, 6 i tako redom do vrednosti 0, kao početne vrednosti. Kod prvog obraćanja keš memoriji ***KEŠ*** nema saglasnosti ni u jednom od 8 ulaza keš memorije ***KEŠ***, jer su svi njeni ulazi nevažeći, pa se iz memorije ***MEM*** dovlači blok podataka. On se smešta u ulaz 0 keš memorije ***KEŠ***, jer LRU brojač ulaza 0 ima vrednost 7. Tom prilikom LRU brojač ulaza 0 postaje 0, LRU brojač ulaz 1 se inkrementira sa 6 na 7, LRU brojač ulaza 2 se inkrementira sa 5 na 6 i tako redom do LRU brojača ulaza 7 koji se inkrementira sa 0 na 1. Kada se sledeći put otkrije da u keš memoriji ***KEŠ*** nema saglasnosti ni u jednom od 8 ulaza keš memorije ***KEŠ***, ponovo se iz memorije ***MEM*** dovlači u keš memoriju ***KEŠ*** blok podataka. On se smešta u ulaz 1 keš memorije ***KEŠ***, jer LRU brojač ulaza 1 ima vrednost 7. Primenom prethodno opisanog načina ažuriranja LRU brojača, LRU brojač ulaza 1 postaje 0, a LRU brojači preostalih 7 ulaza se inkrementiraju. Sadržaj LRU brojača ulaza 2 postaje 7, što znači da će kod prve sledeće nesaglasnosti blok podataka biti prebačen iz memorije ***MEM*** u ulaz 2 keš memorije ***KEŠ***. Po istom postupku se popunjavaju i preostali ulazi keš memorije ***KEŠ***.

1.2.2.3.1.5.2. Strukturna šema

Blok *lru brojači* (slika 51) sadrži LRU brojače LRUCNT0 do LRUCNT7, flip-flop FRQ, komparatore CMP0 do CMP7, koder CD i multipleksere MP1 i MP2.



Slika 51 Blok lru kola

LRU brojači LRUCNT0 do LRUCNT7 i komparatori CMP0 do CMP7 odgovaraju ulazima 0 do 7 keš memorije ***KEŠ***, respektivno. Generisanjem aktivne vrednosti signala **ldLRUCNT** trajanja jedna perioda signal takta pri pojavi signala takta u brojače LRUCNT0 do LRUCNT7 upisuju se početne vrednosti 7 do 0, respektivno. Flip-flopa FRQ (First ReQuest) svojom aktivnom vrednošću označava da se čeka na prvo obraćanje keš memoriji ***KEŠ*** od strane procesora ***CPU***. Ovaj flip-flop se postavla na aktivnu vrednost signalom **mr** kojim se ceo sistem na početku rada asinhrono dovodi u početno stanje. Aktivna vrednost signala logičkog uslova **FRQ** se koristi da se pri prvom pojavljivanju aktivne vrednosti signala **PRQRD** ili **PRQWR** generiše aktivnu vrednost signala **ldLRUCNT** i **clFRQ** trajanja jedna perioda signala takta. Signalom **ldLRUCNT** se upisuju početne vrednosti u LRU brojače, a signalom **clFRQ** flip-flop FRQ se postavlja na neaktivnu vrednost. Time se onemogućava generisanje aktivne vrednosti signala **ldLRUCNT** pri sledećim obraćanjima keš memoriji ***KEŠ*** od strane procesora ***CPU***. Tek kada se ceo sistem asinhrono generisanjem signala **mr** ponovo dovodi u početno stanje i flip-flop FRQ se ponovo postavlja na aktivnu vrednost.

Na ulaze A komparatora CMP0 do CMP7 dovode se signali **LRUCNT0$_{2...0}$** do **LRUCNT7$_{2...0}$** sa izlaza LRU brojača za ulaze 0 do 7 keš memorije ***KEŠ***, respektivno. Na

ulaze B svih komparatora dovede se signali $\mathbf{CNT_{2...0}}$ koji predstavljaju sadržaj LRU brojača ulaza keš memorije ***KEŠ*** kome se pristupa. To je ili ulaz keš memorije ***KEŠ*** u kome je otkrivena saglasnost, pa se ili u taj ulaz keš memorije ***KEŠ*** upisuje ili se iz tog ulaza keš memorije ***KEŠ*** čita, ili je to ulaz keš memorije ***KEŠ*** koji je odabran za zamenu, pa se u njega iz memorije ***MEM*** dovlači blok podataka. Za svaki od 8 komparatora CMP0 do CMP7 aktivna vrednost se javlja samo na jednom od tri izlaza označena sa LSS (LeSS), EQL (EQuaL) i GTR(GreaTeR). Na kome će od tri izlaza svakog od 8 komparatora to biti zavisi od toga da li je sadržaj sa ulaza A komparatora manji, jednak ili veći od sadržaja sa ulaza B komparatora. Aktivna vrednost se javlja na izlazima LSS komparatora koji odgovaraju ulazima keš memorije ***KEŠ*** čiji LRU brojači imaju manju vrednost od vrednosti LRU brojača ulaza keš memorije ***KEŠ*** kome se pristupa. Pri pojavi aktivne vrednosti signal **adjLRUCNT** trajanja jedna perioda signala takta na ulazima INC odgovarajućih LRU brojača javlja se aktivna vrednost trajanja jedna perioda signala takta dok su na ulazima CL neaktivne vrednosti, pa se pri pojavi signal takta inkrementiraju odgovarajući LRU brojači. Aktivna vrednost se javlja na izlazu EQL samo onog komparatora koji odgovara LRU brojaču ulaza keš memorije ***KEŠ*** kome se pristupa. S obzirom da se radi o komparatoru ulaza keš memorije ***KEŠ*** kome se pristupa, na ulazu B kao signali $\mathbf{CNT_{2...0}}$ selektovani su kroz multiplekser MP1 signali sa LRU brojača ulaza keš memorije ***KEŠ*** kome se pristupa. S toga su kod ovog komparatora na ulazima A i B iste vrednosti, pa je i signal na izlazu EQL aktivan. Pri pojavi aktivne vrednosti signala **adjLRUCNT** na ulazu CL odgovarajućeg LRU brojača javlja se aktivna vrednost, dok je na ulazu INC neaktivna vrednost, pa se pri pojavi signala takta briše i postavlja na nulu odgovarajući LRU brojač. Aktivna vrednost se javlja na izlazima GTR komparatora koji odgovaraju ulazima keš ***KEŠ*** memorije čiji LRU brojači imaju veće vrednosti od vrednosti LRU brojača ulaza keš memorije ***KEŠ*** kome se pristupa. Pri pojavi aktivne vrednosti signala **adjLRUCNT** na ulazima INC i CL odgovarajućih LRU brojača se neaktivne vrednosti, pa se pri pojavi signala takta ne menjaju vrednosti odgovarajućih LRU brojača.

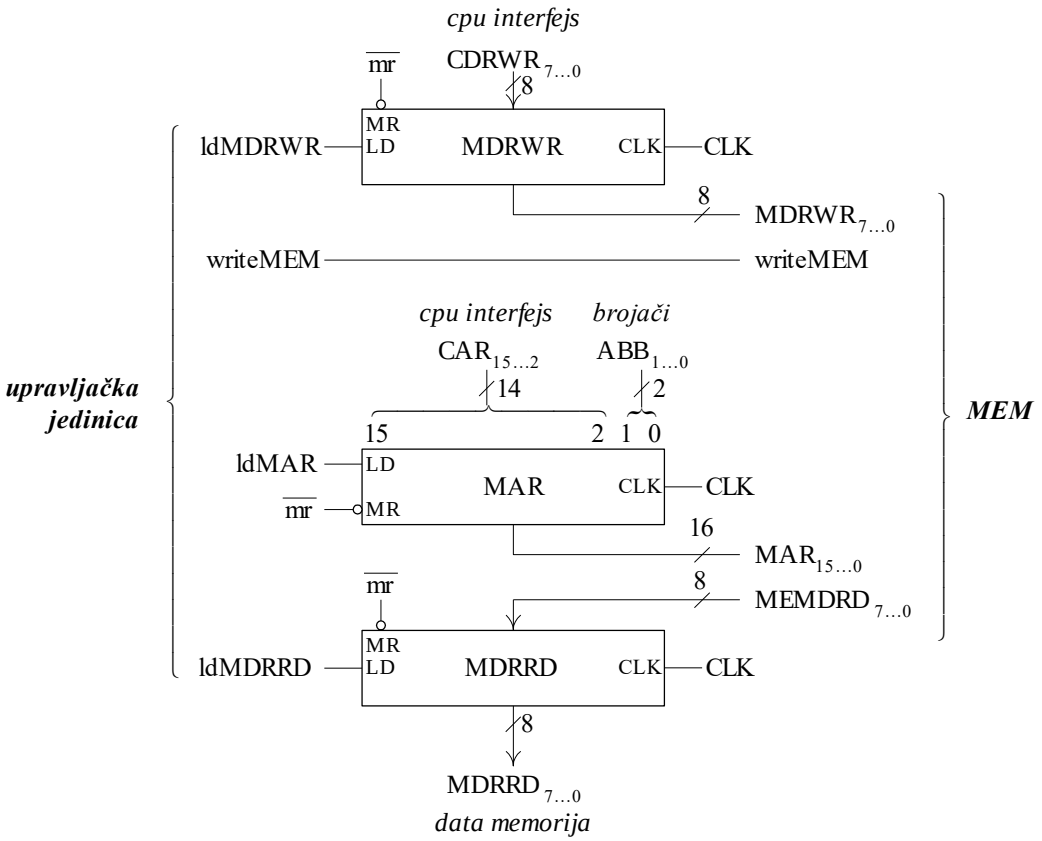
Multiplekser MP1 služi za formiranje signala $\mathbf{CNT_{2...0}}$ koji se kao signali selektovanog LRU brojača ulaza keš memorije ***KEŠ*** kome se pristupa vode na ulaze B komparatora CMP0 do CMP7. Na ulaze 0 do 7 multipleksera MP1 vode se LRU brojači $LRUCNT0_{2...0}$ do $LRUCNT7_{2...0}$. Sadržaj jednog od ovih brojača se selektuje upravljačkim signalom $\mathbf{mxLRUCNT_{2...0}}$ i pojavljuje kao signali $\mathbf{CNT_{2...0}}$. Signali $\mathbf{mxLRUCNT_{2...0}}$ predstavljaju binarnu vrednost ulaza keš memorije ***KEŠ*** kome se pristupa.

Koder CD služi za formiranje signala $\mathbf{LRU_{2...0}}$ koji predstavljaju binarnu vrednost ulaza keš memorije ***KEŠ*** u koji se dovlači blok podataka iz memorije ***MEM***. Na ulaze 0 do 7 kodera CD dovede se signali sa izlaza odgovarajućih logičkih I kola. Na ulaze I kola čiji se izlaz vodi na ulaz 0 kodera CD vezani su signali $\mathbf{LRUCNT0_{2...0}}$ sa izlaza LRU brojača ulaza 0 keš memorije ***KEŠ***. Na ulaze I kola čiji se izlaz vodi na ulaz 1 kodera CD vezani su signali $\mathbf{LRUCNT1_{2...0}}$ sa izlaza LRU brojača ulaza 1 keš memorije ***KEŠ***. Na identičan način se na ulaze I kola čiji se izlazi vode na ulaze 2 do 7 kodera CD vezani signali sa izlaza LRU brojača ulaza 2 do 7 keš memorije ***KEŠ***. S obzirom da je u nekom trenutku vrednost 7 samo u jednom od LRU brojača, i to onome koji odgovara ulazu keš memorije ***KEŠ*** kome se najmanje skoro pristupilo, samo na ulazima jednog od 8 logičkih I kola su sva tri ulazna signala aktivni, samo na njegovom izlazu je aktivna vrednost i samo na jednom od ulaza kodera CD je aktivna vrednost. Izlazni signali $\mathbf{LRU_{2...0}}$ predstavljaju binarnu vrednost ulaza kodera CD na kome je aktivna vrednost. To je binarna vrednost ulaza keš memorije ***KEŠ*** čiji LRU brojač ima vrednost 7. S obzirom da vrednost 7 ima LRU brojač ulaza keš memorije ***KEŠ*** kome se najmanje skoro pristupilo, signali $\mathbf{LRU_{2...0}}$ predstavljaju binarnu vrednost ulaza keš memorije ***KEŠ*** koji se zamenjuje novim blokom i u koji se dovlači blok podataka iz memorije ***MEM***.

Multiplekser MP2 služi za formiranje signala **mxLRUCNT$_{2...0}$** koji predstavljaju binarnu vrednost ulaza keš memorije ***KEŠ*** kome se pristupa. Na ulaze 0 i 1 multipleksera MP2 dovode se signali **LRU$_{2...0}$** koji predstavljaju broj ulaza keš memorije ***KEŠ*** u koji se dovlači blok podataka i signali **AMADR$_{2...0}$** koji predstavljaju broj ulaza keš memorije ***KEŠ*** u kome je otkrivena saglasnost, respektivno. Na izlazima se formiraju signali **mxLRUCNT$_{2...0}$** koji predstavljaju binarnu vrednost ulaza keš memorije ***KEŠ*** kome se pristupa. Ovi signali se koriste kao upravljački signali multipleksera MP1 za selekciju LRU brojača ulaza keš memorije ***KEŠ*** kome se pristupa i formiranje signala **CNT$_{2...0}$**. U slučaju kada nema saglasnosti broja ulaza keš memorije ***KEŠ*** kome se pristupa je broja ulaza keš memorije ***KEŠ*** u koji se dovlači blok podataka i određen je signalima **LRU$_{2...0}$**. U slučaju kada ima saglasnosti broj ulaza keš memorije ***KEŠ*** kome se pristupa je broj ulaza u kome je otkrivena saglasnost i određen je signalima **AMADR$_{2...0}$** . S toga se neaktivnom i aktivnom vrednošću signala saglasnosti **H/M** kroz multiplekser MP2 kao signali **mxLRUCNT$_{2...0}$** propuštaju signali **LRU$_{2...0}$** i **AMADR$_{2...0}$**, respektivno.

1.2.2.3.1.6. Blok mem interfejs

Blok *mem interfejs* (slika 52) sadrži registre MAR, MDRWR i MDRRD.



Slika 52 Blok mem interfejs

Registar MAR (Memory Address Register) služi za čuvanje ili adresa lokacija memorije ***MEM*** sa kojih treba očitati podatke u slučaju dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** zbog toga što je za operaciju čitana utvrđeno da nema saglasnosti ili adrese lokacije memorije ***MEM*** u koju treba upisati podatak u slučaju operacije upisa. Adresa se upisuje u registar MAR pri pojavi signala takta i aktivnoj vrednosti signala **ldMAR**. Adresa koja se upisuje formira se iz dva dela i to na sledeći način:

- *Bitovi 15 do 2*. Kao bitovi 15 do 2 koriste se signali $\mathbf{CAR_{15...2}}$ registra CAR.
- *Bitovi 1 do 0*. Kao bitovi 1 do 0 koriste se signali $\mathbf{ABB_{1...0}}$ brojača ABB.

Izlazi registra MAR se vode kao signali $\mathbf{MAR_{15...0}}$ na adresne linije $A_{15...0}$ memorije ***MEM***.

Registar MDRWR (Memory Data Register for WRite) služi za čuvanje podatka koji treba upisati u memoriju ***MEM*** u slučaju operacije upisa. Podatak za upis se dovodi na ulaze registra MDRWR kao signali $\mathbf{CDRWR_{7...0}}$ i upisuje u registar MDRWR na signal takta ukoliko je aktivan signal **ldMDRWR**. Izlazi registra MDRWR se kao signali $\mathbf{MDRWR_{7..0}}$ vode na ulazne linije podataka $DI_{7..0}$ memorije ***MEM***.

Registar MDRRD (Memory Data Register for ReaD) služi za čuvanje podatka koji je očitan iz memorije ***MEM*** pri dovlačenju bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** u slučaju operacije čitanja i nesaglasnosti u keš memoriji ***KEŠ***. Ovaj podatak se kao signali $\mathbf{MEMDRD_{7...0}}$ dovodi sa izlaznih linija podataka $DO_{7...0}$ memorije ***MEM*** na ulaze registra MDRRD i upisuje u registar MDRRD na signal takta ukoliko je aktivan signal **ldMDRRD**. Izlazi registra MDRRD se kao signali $\mathbf{MDRRD_{7...0}}$ vode u blok *data memorija* radi upisa u memorjski modul DATA.

1.2.2.3.2. Upravljačka jedinica

Upravljačka jedinica keš memorije ***KEŠ*** služi za generisanje upravljačkih signala prema algoritmu asocijativnog preslikavanja. U ovom poglavlju se daju dijagram toka operacija, algoritam generisanja upravljačkih signala, vremenski oblici signala i struktura ***upravljačke jedinice***.

1.2.2.3.2.1. Dijagram toka operacija

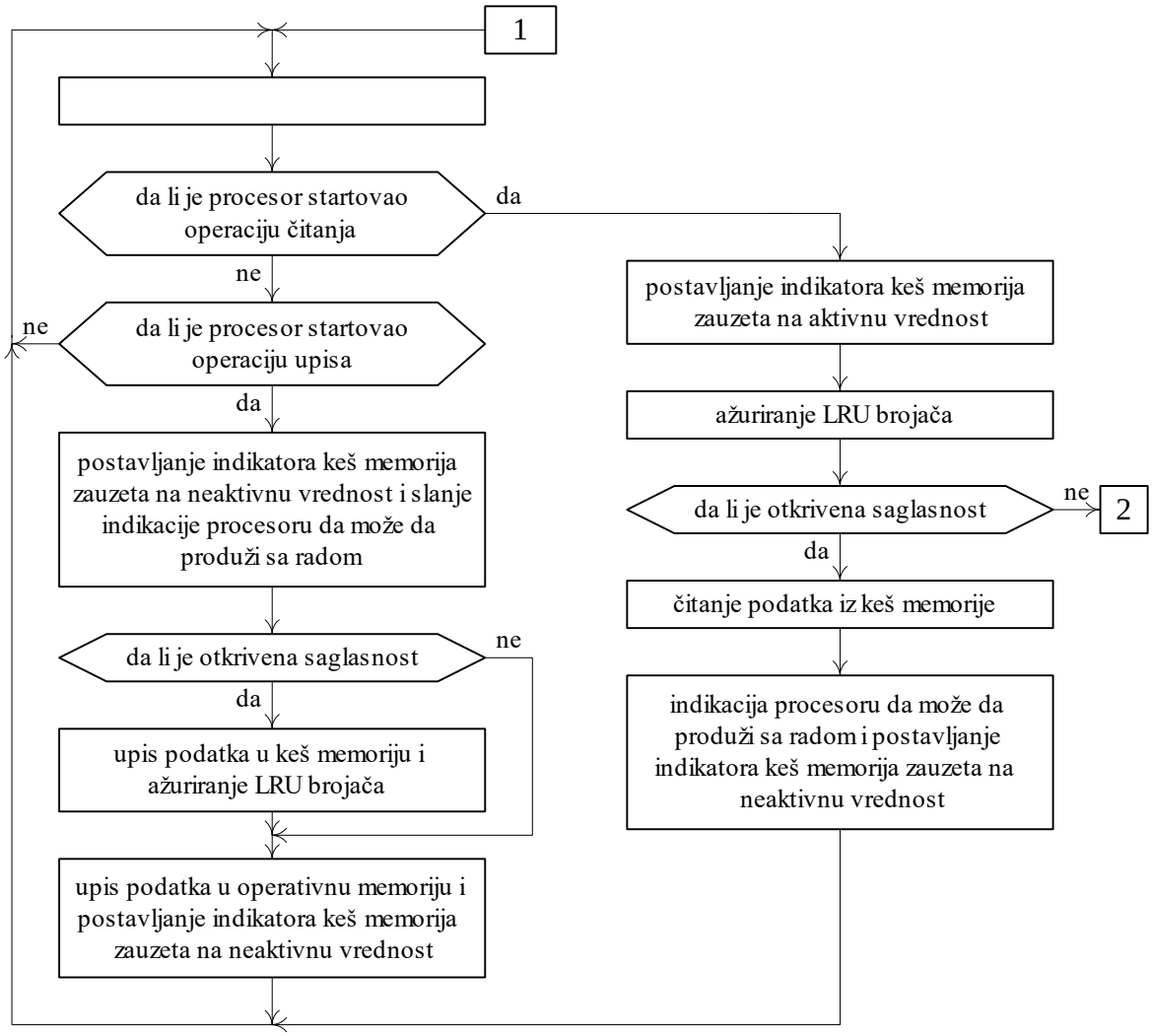
Dijagram toka operacija je dat na slikama 53 i 54.

U početnom stanju, upravljačka jedinica keš memorije ***KEŠ*** čeka pojavu signala startovanja operacije čitanja **PRQRD** ili signala startovanja operacije upisa **PRQWR**. U slučaju startovanja i operacije čitanja i operacije upisa procesor šalje adresu operativne memorije koja se upisuje u prihvatni registar adrese keš memorije, dok u slučaju startovanja operacije upisa procesor šalje podatak koji se upisuje u prihvatni registar podatka keš memorije. Kao reakcija na pojavu signala **PRQRD** ili **PRQWR**, keš memorija šalje procesoru indikaciju da je zauzeta i da do daljeg nije spremna da prihvaru novu operaciju čitanja ili upisa. Posle toga keš memorija počinje sa izvršavanjem odgovarajuće operacije, pri čemu sledeći koraci zavise od toga da li se radi o operaciji čitanja ili upisa.

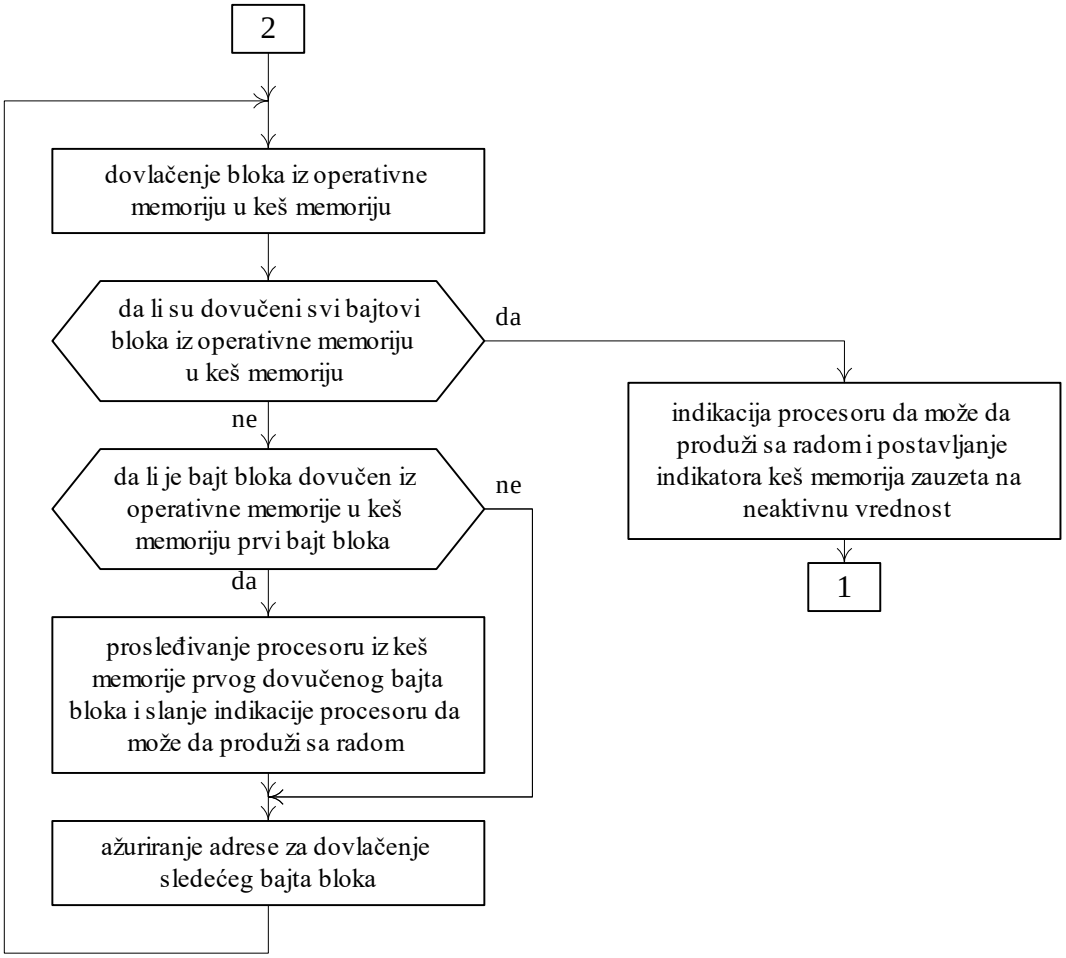
Ako se radi o operaciji upisa keš memorija odmah šalje indikaciju o završetku operacije. Zatim se vrši provera da li ima saglasnosti. U slučaju da postoji saglasnost podatak se upisuje u keš memoriju i LRU brojači ažuriraju. U suprotnom slučaju ovaj korak se preskače. Bez obzira na to da li postoji saglasnost ili ne vrši se upis podatka u operativnu memoriju. Nakon toga keš memorija šalje procesoru indikaciju da nije više zauzeta i da je nadalje spremna da prihvaru novu operaciju čitanja ili upisa.

Ako se radi o operacija čitanja prvo se LRU brojači ažuriraju. Posle toga se vrši provera da li ima saglasnosti. U slučaju da postoji saglasnost podatak se čita iz keš memorije i šalje procesoru. Nakon toga keš memorija šalje indikaciju o završetku operacije i indikaciju da nije više zauzeta i da je nadalje spremna da prihvaru novu operaciju čitanja ili upisa. U slučaju da saglasnost ne postoji prelazi se na dovlačenje bloka podataka iz operativne memorije u keš memoriju, pri čemu se prvo dovlači onaj bajt podatka iz bloka čije je čitanje procesor zahtevao. Očitani bajt podatka se istovremeno upisuje u keš memoriju i šalje procesoru

zajedno sa indikacijom o završetku operacije. Posle toga iz operativne memorije se čitaju i u keš memoriju upisuju i preostala tri bajta bloka. Na kraju keš memorija šalje procesoru indikaciju da nije više zauzeta i da je nadalje spremna da prihvaru novu operaciju čitanja ili upisa.



Slika 53 Dijagram toka operacija (prvi deo)



Slika 54 Dijagram toka operacija (drugi deo)

1.2.2.3.2.2. Algoritam generisanja upravljačkih signala

Algoritam generisanja upravljačkih signala je formiran na osnovu strukture ***operacione jedinice*** (poglavlje 1.2.1.3.1) i dijagrama toka operacija (poglavlje 1.2.1.3.2.1). Notacija koja se koristi je identična sa notacijom iz poglavlja 1.2.1.1.2.2.

Algoritam generisanja upravljačkih signala je dat u daljem tekstu.

$step_0$: *br* (*if* (**PRQRD** + **PRQWR**) *then* $step_1$ *else* $step_0$)

! U koraku $step_0$ se čeka da iz procesora ***CPU*** preko bloka *cpu interfejs* stigne signal startovanja operacije čitanja **PRQRD** ili signal startovanja operacije upisa **PRQWR**. Ako se pojavi signal **PRQRD** na signal takta se adresa upisuje u registar CAR bloka *cpu interfejs* i flip-flop CRD bloka *cpu interfejs* se postavlja na aktivnu vrednost. Ako se pojavi signal **PRQWR** na signal takta se adresa upisuje u registar CAR, podatak u registar CDRWR bloka *cpu interfejs* i flip-flop CWR bloka *cpu interfejs* se postavlja na aktivnu vrednost. Postavljanjem jednog od flip-flopova CRD ili CWR na aktivnu vrednost i signal zauzetosti keš memorije **BUSY** bloka *cpu interfejs* postaje aktivan. Ako se pojavi bilo koji od signala **PRQRD** ili **PRQWR** na signal takta se prelazi na korak $step_1$. U suprotnom slučaju se ostaje u koraku $step_0$.

$step_1$: *if* ((**CRD** · **H/M**), **ldCDRRD**),
if ((**CRD** · $\overline{\textbf{H/M}}$ + **CWR**), **ldABB**),
if (((**CRD** + **CWR**) · **H/M**), **adjLRUCNT**),
if (**CWR**, **CRP**),
br (*if* **CWR** *then* $step_3$ *else* (*if* **H/M** *then* $step_2$ *else* $step_5$))

! U korak $step_1$ može da se dođe jedino iz koraka $step_0$ i to jedino kada se startuje operacija čitanja ili operacija upisa. U koraku $step_1$ se i za operaciju čitanja i za operaciju upisa vrši provera saglasnosti i na osnovu

toga formira odgovarajuća vrednosta signala **H/M** bloka *tag memorija*. Aktivna vrednost signala **H/M** označava da je otkrivena saglasnost. U slučaju operacije čitanja signal **CRD** bloka *cpu interfejs* je aktivan. Ako je formirana aktivna vrednost signala **H/M** generiše se aktivna vrednost signala **ldCDRRD**, pa se na signal takta podatak očitan iz memorijskog modula DATA bloka *data memorija* upisuje u registar CDRRD bloka *cpu interfejs*. U slučaju operacije upisa signal **CWR** bloka *cpu interfejs* je aktivan. Ukoliko je u pitanju operacija upisa ili ako je u pitanju operacija čitanja, a pritom je signal **H/M** na neaktivnoj vrednosti, generiše se aktivna vrednost signala **ldABB** pa se na signal takta dva najniža bita generisane adrese iz registra CAR bloka *cpu interfejs* upisuju u brojač ABB bloka *brojači*. U slučaju operacije čitanja ili operacije upisa ukoliko je signal **H/M** na aktivnoj vrednosti, pristupa se jednom od ulaza memorijskog modula DATA bloka *data memorija*. S toga se tada generiše aktivna vrednost signala **adjLRUCNT**, pa se na signal takta ažuriraju LRU brojači bloka *lru kola*. U slučaju operacije upisa generiše se aktivna vrednost signala **CRP** bloka *cpu interfejs*. Time se omogućava procesoru ***CPU*** da ranije nastavi sa radom iako operacija upisa nije završena. Međutim, signal zauzetosti **BUSY** bloka *cpu interfejs* ostaje aktivan sve dok se operacija upisa ne završi. Aktivna vrednost signala **BUSY** sprečava procesor ***CPU***, kome je dozvoljeno da ranije nastavi sa radom, da eventualno startuje novu operaciju čitanja ili upisa. Kada tekuća operacije upisa bude završena flip-flop CWR će biti postavljen na neaktivnu vrednost, signal **BUSY** će postati neaktivan, pa će procesor ***CPU*** tek tada moći da staruje novu operaciju čitanja ili upisa. U slučaju operacije upisa iz koraka $step_1$ se prelazi na korak $step_3$. U slučaju operacije čitanja se u zavisnosti od toga da li je signala **H/M** aktivan ili neaktivan prelazi na korak $step_2$ ili korak $step_5$, respektivno.

$step_2$: **CRP**, **clCRD**,
br $step_0$

! U korak $step_2$ se dolazi jedino iz koraka $step_1$ i to samo onda kada je u koraku $step_1$ prilikom operacije čitanja otkrivena saglasnost. U ovom koraku se bezuslovno generišu aktivne vrednosti signala **CRP** i **clCRD** bloka *cpu interfejs*. Signalom **CRP** se signalizira procesoru ***CPU*** da je očitani podatak raspoloživ i da procesor ***CPU*** može da nastavi sa radom. Signalom **clCRD** se omogućuje da se na signal takta upiše neaktivna vrednost u flip-flop CRD, signal **BUSY** postavi na neaktivnu vrednost, čime se procesoru ***CPU*** omogućuje da startuje novu operaciju čitanja ili upisa u keš memoriji ***KEŠ***. Na signal takta se bezuslovno prelazi iz koraka $step_2$ na korak $step_0$.

$step_3$: **ldMDRWR**, **ldMAR**,
if (**H/M**, **writeDATA**),
br $step_4$

! U korak $step_3$ se dolazi jedino iz koraka $step_1$ u slučaju da je u pitanju operacija upisa. U ovom koraku se bezuslovno generišu aktivne vrednosti signala **ldMDRWR** i **ldMAR** kojima se podatak koji treba upisati u memoriju ***MEM*** i adresa memorije ***MEM*** na signal takta upisuju u registre MDRWR i MAR bloka *mem interfejs*, respektivno. Ukoliko je aktivna vrednost signala **H/M** bloka *tag memorija* generiše se i aktivna vrednost signala **writeDATA** bloka *data memorija*. Time se u slučaju kada je za operaciju upisa otkrivena saglasnost upisuje nova vrednost u memorijski modul DATA. Iz koraka $step_3$ se bezuslovno prelazi na korak $step_4$.

$step_4$: **writeMEM**, **incMEMACC**,
if (**MEMFC**, **clCWR**),
br (*if* **MEMFC** *then* $step_0$ *else* $step_4$)

! U korak $step_4$ se dolazi samo iz koraka $step_3$. U ovom koraku se bezuslovno generišu aktivne vrednosti signala **writeMEM** bloka *mem interfejs* i **incMEMACC** bloka *brojači*. Aktivnom vrednošću signala **writeMEM** se sadržaj registra MDRWR upisuje u memoriju MEM na adresi određenoj sadržajem registra MAR. Aktivnom vrednošću signala **incMEMACC** se pri pojavi signala takta inkrementira sadržaj brojača MEMACC koji određuje vreme pristupa memoriji ***MEM***. Pri ulazu u korak $step_4$ brojač MEMACC ima vrednost nula, pa signal **MEMFC** bloka *brojači* ima neaktivnu vrednost. Na deveti signal takta na koji sadržaj brojača MEMACC postaje devet, signal **MEMFC** postaje aktivan. Pri aktivnoj vrednosti signala **MEMFC** i signal **clCWR** bloka *cpu interfejs* postaje aktivan. Aktivnom vrednošću signala **clCWR** trajanja jedna perioda signala takta se na signal takta upisuje neaktivna vrednost u flip-flop CWR, signal **BUSY** bloka *cpu interfejs* se postavlja na neaktivnu vrednost tako da će procesor ***CPU*** moći da startuje novu operaciju čitanja ili upisa u keš memoriji ***KEŠ***. Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku $step_4$, dok se pri njegovoj aktivnoj vrednosti na signal takta prelazi na korak $step_0$.

$step_5$: **ldMAR**, *if* (**CNTBB1**, **CRP**),
br $step_6$

! U korak $step_5$ se dolazi iz koraka $step_1$ u kome je za operaciju čitanja utvrđeno da nema saglasnosti i da treba preći na dovlačenje bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ***. Dovlačenje jednog bajta podatka se realizuje u koracima $step_5$, $step_6$ i $step_7$. Prilikom dovlačenja bloka podataka četiri puta mora da se prođe kroz ova tri koraka, pošto je veličina bloka četiri bajta. Ti prolasci se u tekstu nazivaju nulti, prvi, drugi i treći. Pored toga usvojeno je da se najpre dovuče bajt bloka podatka za koji je generisana adresa, da se taj bajt podatka odmah prosledi procesoru ***CPU***, da se procesoru ***CPU*** omogući da produži sa radom, da se zatim dovuku i preostala tri bajta podatka bloka i da se tek tada procesoru ***CPU*** signalizira da keš memorija ***KEŠ*** nije više zauzeta. Za realizaciju dovlačenja koriste se brojači ABB i CNTBB bloka *brojači* oba po modulu četiri. Brojač ABB svojom vrednošću određuje za prolaze 0 do 3 adrese bajtova u bloku, počev od bajta bloka za koji je generisana adresa. Brojač CNTBB svojom vrednošću određuje prolaze 0 do 3, počev od prolaza 0. Signali **CNTBB0**, **CNTBB1** i **CNTBB3** bloka *brojači* su signali dekodivanih stanja 0, 1 i 3 brojača CNTBB i svojim aktivnim vrednostima određuje da se radi o prolascima 0, 1 i 3, respektivno. U koraku $step_5$ se bezuslovno generiše aktivna vrednost signala **ldMAR**, čime se na signal takta u registar MAR bloka *mem interfejs* upisuje adresa memorije ***MEM***. U prolasku 0 to je adresa bajta podatka za koju je procesor ***CPU*** u keš memoriji ***KEŠ*** startovao operaciju čitanja. U prolasku 0 se u koraku $step_6$ podatak čita iz memorije ***MEM***, a u koraku $step_7$ upisuje u memorijski modul DATA bloka *data memorija* i registar CDRRD bloka *cpu interfejs*. U koraku $step_7$ se još i inkrementiraju brojači ABB i CNTBB. Prilikom prolaska 1 kroz korak $step_5$ ponovo se generiše aktivna vrednost signala **ldMAR**, čime se na signal takta u registar MAR upisuje adresa memorije ***MEM*** sa koje treba očitati sledeći bajt bloka. Međutim, sada je signal **CNTBB1** aktivan, pa se generiše aktivna vrednost signala **CRP** bloka *cpu interfejs*. Ovim signalom se bajt podatka očitan iz memorije ***MEM*** u prolasku 1 prosleđuje iz registar CDRRD bloka *cpu interfejs* u procesor ***CPU***. Pored toga aktivna vrednost signala **CRP** omogućava procesoru ***CPU*** da ranije nastavi sa radom iako preostala tri bajta bloka tek treba da se dovuku iz memorije ***MEM*** u keš memoriju ***KEŠ*** u prolascima 1, 2 i 3. Međutim, signal zauzetosti **BUSY** bloka *cpu interfejs* ostaje aktivan sve dok se preostala tri bajta bloka ne dovuku. Aktivna vrednost signala **BUSY** sprečava procesor ***CPU*** kome je dozvoljeno da ranije nastavi sa radom, da eventualno startuje novu operaciju čitanja ili upisa u keš memoriji *KEŠ*. Iz ovog koraka se bezuslovno prelazi na korak $step_6$.

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

! U korak $step_6$ se dolazi iz koraka $step_5$. U koraku $step_6$ se bezuslovno generiše aktivna vrednost signala **incMEMACC** bloka *brojači* čime se na signal takta inkrementira sadržaj brojača MEMACC koji određuje vreme pristupa memoriji ***MEM***. Pri ulazu u korak $step_6$ brojač MEMACC ima vrednost nula, pa signal **MEMFC** bloka *brojači* ima neaktivnu vrednost. Na deveti signal takta na koji sadržaj brojača MEMACC postaje devet, signal **MEMFC** postaje aktivan. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDRRD** bloka *mem interfejs*, čime se obezbeđuje da se na signal takta u registar MDRRD upiše vrednost očitana iz memorije ***MEM***. Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku $step_6$ pošto nije završena operacija očitavanja jednog bajta podatka iz memorije ***MEM***. Pri aktivnoj vrednosti signala **MEMFC** se na signal takta prelazi iz koraka $step_6$ na korak $step_7$.

$step_7$: **mxADATA**, **mxDIDATA**, **writeDATA**, **incABB**, **incCNTBB**,
if (**CNTBB0**, **mxDATA**, **ldCDRRD**),
if (**CNTBB3**, **writeTAG**, **writeV**, **adjLRUCNT, clCRD**),
br (*if* **CNTBB3** *then* $step_0$ *else* $step_5$)

! U korak $step_7$ se dolazi samo iz koraka $step_6$. U koraku $step_7$ se bezuslovno generišu aktivne vrednosti signala **mxADATA**, **mxDIDATA** i **writeDATA** bloka *data memorija* i **incABB** i **incCNTBB** bloka *brojači*. Aktivnom vrednošću signala **mxADATA** se na adresne linije memorijskog modula DATA dovodi adresa, aktivnom vrednošću signala **mxDIDATA** se na ulazne linije podataka memorijskog modula DATA dovodi bajt podatka očitan iz memorije ***MEM*** i aktivnom vrednošću signala **writeDATA** se vrši upis u memorijski modul DATA. Aktivnim vrednostima signala **incABB** i **incCNTBB** se na signal takta inkrementiraju sadržaji brojača ABB i CNTBB, respektivno. Prilikom prolaska 0 kroz korak $step_7$ brojač CNTBB ima vrednost nula, pa je signal **CNTBB0** bloka *brojači* aktivan. Pri aktivnoj vrednosti signala **CNTBB0** generišu se aktivne vrednosti signala **mxDATA** i **ldCDRRD**. Pri aktivnoj vrednosti signala **mxDATA** se u bloku *data memorija* selektuje bajt podatka očitan iz memorije ***MEM***. Ovaj podatak se vodi na ulazne linije registra CDRRD bloka *cpu interfejs* u koji se i upisuje pri aktivnoj vrednosti signala **ldCDRRD** i pojavi signala takta . Prilikom prolaska 3 kroz korak $step_7$ brojač CNTBB ima vrednost tri, pa je signal **CNTBB3** bloka *brojači* aktivan. Pri aktivnoj vrednosti signala **CNTBB3** generišu se aktivne vrednosti signala **wrTAG** bloka *tag memorijai*, **writeV** bloka *indikatoei*, **adjLRUCNT** bloka *lru brojači* i **clCRD** bloka *cpu interfejs*. Aktivnom vrednošću signala **writeTAG** se u memorijski modul TAG upisuje broja bloka memorije ***MEM*** koji je dovučen iz memorije ***MEM*** u odgovarajući ulaz memorijskog modula DATA bloka *data memorija*. Aktivnom vrednošću signala **writeV** se na signal takta odgovarajući flip-flop V postavlja na aktivnu vrednost. Aktivnom vrednošću signala **adjLRUCNT** se na signal

takta ažuriraju LRU brojači bloka *lru kola*. Aktivnom vrednošću signala **clCRD** se upisuje neaktivna vrednost u flip-flop CRD i signal **BUSY** bloka *cpu interfejs* se postavlja na neaktivnu vrednost, tako da će procesor ***CPU*** moći da startuje novu operaciju čitanja ili upisa. Iz koraka $step_7$ se pri neaktivnoj vrednosti signala **CNTBB3** bloka *brojači* prelazi na korak $step_5$, a pri njegovoj aktivnoj vrednosti na korak $step_0$.

1.2.2.3.2.3. Vremenski oblici signala

U keš memoriji ***KEŠ*** mogu da se izvršavaju operacije čitanja i upisa. Izvršavanje svake operacije se sastoji od nekoliko aktivnosti čija realizacija zahteva jedan ili više koraka. Te aktivnosti i koraci u kojima se one realizuju su:

1. *Čekanje signala startovanja operacije čitanja ili upisa*. Ova aktivnost se realizuje u koraku $step_0$.
2. *Provera da li ima saglasnosti*. Ova aktivnost se realizuje u koraku $step_1$.
3. *Čitanje bajta podatka*. Ova aktivnost se realizuje u koraku $step_1$ i koraku $step_7$
4. *Ažuriranje LRU brojača*. Ova aktivnost se realizuje u koraku $step_1$.
5. *Slanje indikacije da procesor **CPU** može da produži sa radom*. Ova aktivnost se realizuje u koraku $step_2$ u slučaju operacija čitanja i ima saglasnosti, u koraku $step_5$ u slučaju operacija čitanja i nema saglasnosti i u koraku $step_1$ u slučaju operacija čitanja.
6. *Upis bajta podatka u keš memoriju **KEŠ***. Ova aktivnost se realizuje u koraku $step_3$.
7. *Upis bajta podatka u memoriju **MEM***. Ova aktivnost se realizuje u koraku $step_4$.
8. *Dovlačenje bajtova bloka*. Ova aktivnost se realizuje u koracima $step_5$, $step_6$ i $step_7$.

Vremenski oblici signala za operacije čitanja i upisa za različite slučajeve u kojima keš memorija ***KEŠ*** može da se nađe dobijaju se kombinovanjem vremenskih oblika signala za osam navedenih aktivnosti.

Vremenski oblici signala za obe operacije i to za različite situacije u kojima pri tome keš memorija ***KEŠ*** može da se nađe se daju u daljem tekstu.

1.2.2.3.2.3.1. Operacija čitanja

U slučaju operacije čitanja mogu da se jave dva slučaja:

- nema saglasnosti i
- ima saglasnosti.

Vremenski oblici signala za oba slučaja dati su u daljem tekstu.

Vremenski oblici signala za slučaj kada se u keš memoriji ***KEŠ*** izvršava operacija čitanja i pri tome nema saglasnosti dati su na slici 55.

Keš memorija ***KEŠ*** u koraku $step_0$ čeka pojavu aktivne vrednosti signala startovanja operacije čitanja **PRQRD** koji preko bloka *cpu interfejs* dolazi iz procesora ***CPU***. Pri njenoj pojavi na signal takta se u registar CAR bloka *cpu interfejs* upisuje adresa, flip-flop CRD bloka *cpu interfejs* se postavlja na aktivnu vrednost i prelazi se u korak $step_1$. Postavljanjem flip-flopa CRD na aktivnu vrednost generišu se aktivne vrednosti signala **CRD** i **BUSY** bloka *cpu interfejs*.

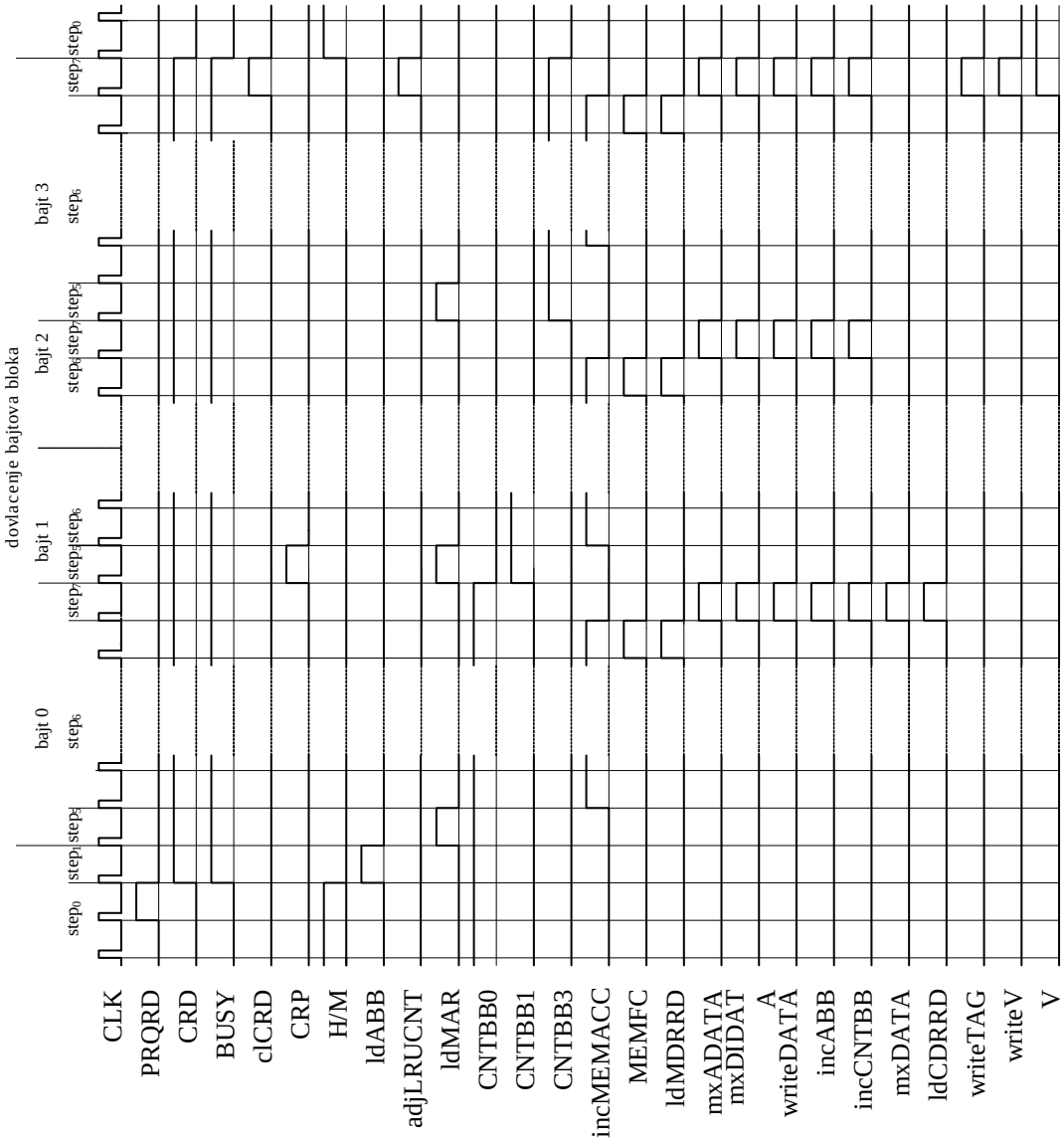
U koraku $step_1$ se generišu aktivne vrednosti signala **ldABB** bloka *brojači*. Pošto je signal saglasnosti **H/M** bloka *tag memorija* neaktivan na signal takta se prelazi na korak $step_5$.

Kroz korake $step_5$, $step_6$ i $step_7$ se prolazi četiri puta, za svaki bajt podatka po jednom. Prilikom prolazaka 0, 1 i 3 generišu se aktivne vrednosti signala **CNTBB0**, **CNTBB1** i **CNTBB3** bloka *brojači*, respektivno.

U koraku $step_5$ se generiše aktivna vrednost signala **ldMAR** bloka *mem interfejs*, dok se samo za prolazak 1 generiše i aktivna vrednost signala **CRP** bloka *cpu interfejs*. Ovim signalom se bajt podatka očitan iz memorije ***MEM*** u prolazu 0 prosleđuje iz registra CDRRD

bloka *cpu interfejs* u procesor ***CPU*** i procesoru ***CPU*** šalje indikacija da može da produži sa radom. U koraku $step_5$ se ostaje samo jedna perioda signala takta i na signal takta se prelazi na korak $step_6$.

U koraku $step_6$ se generiše aktivna vrednost signala **incMEMACC** bloka *brojači* trajanja deset perioda signala takta. Na deveti signal takta postaje aktivan signal **MEMFC** bloka *brojači*. Tada se generiše aktivna vrednost i signala **ldMDRRD** bloka *mem interfejs*, pa se podatak očitan iz memorije ***MEM*** upisuje u registar MDRRD. Pri aktivnoj vrednosti signala **MEMFC** se na signal takta prelazi u korak $step_7$.

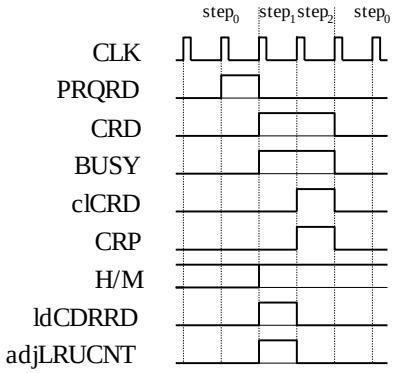


Slika 55 Vremenski oblici signala za slučaj operacija čitanja i nema saglasnosti

U koraku $step_7$ se generišu aktivne vrednosti signala **mxADATA**, **mxDIDATA** i **writeDATA** bloka *data memorija* čime se omogućuje upis podatka iz registra MDRRD bloka *mem interfejs* u memorijski modul DATA. Pored toga generišu se i aktivne vrednosti signala **incABB** i **incCNTBB** bloka *brojači*, pa se na signal takta inkrementiraju brojači ABB i CNTBB. Pri prolazu 0 aktivan je signal **CNTBB0** bloka *brojači*, pa se generišu i aktivne vrednost signala **mxDATA** bloka *data memorija* i **ldCDRRD** bloka *cpu interfejs*. Ovim signalima se omogućuje i upisivanje podatka iz registra MDRRD u registar CDRRD. Pri

prolazu 3 aktivan je signal **CNTBB3** bloka *brojači*, pa se generišu i aktivne vrednosti signala **writeTAG** bloka *tag memorija,* **writeV** bloka *indikatori,* **adjLRUCNT** bloka *lru kola* i **clCRD** bloka *cpu interfejs*. Signalom **writeTAG** se u memorijski modul TAG upisuje broj bloka memorije ***MEM*** koji je dovučen iz memorije ***MEM*** u keš memoriju ***KEŠ***, signalom **writeV** odgovarajući indikator V se postavlja na aktivnu vrednost, signalom **adjLRUCNT** ažuriraju se LRU brojači bloka *lru kola* i signalom **clCRD** flip-flop CRD se postavlja na neaktivnu vrednost. Kada signal **CRD** bloka *cpu interfejs* postane neaktivan i signal **BUSY** bloka *cpu interfejs* postane neaktivan, čime se procesoru ***CPU*** šalje indikacija da keš memorija ***KEŠ*** nije više zauzeta. Na signal takta se pri aktivnoj vrednosti signala **MEMFC** bloka *brojači* u zavisnosti od toga da li signal **CNTBB3** bloka *brojači* neaktivan ili aktivan prelazi u korak $step_5$ i $step_0$, respektivno.

Vremenski oblici signala za slučaj kada se u keš memoriji ***KEŠ*** izvršava operacija čitanja i pri tome ima saglasnosti dati su na slici 56.



Slika 56 Vremenski oblici signala za slučaj operacija čitanja i ima saglasnosti

Keš memorija ***KEŠ*** u koraku $step_0$ čeka pojavu aktivne vrednosti signala startovanja operacije čitanja **PRQRD** koji preko bloka *cpu interfejs* dolazi iz procesora ***CPU***. Pri njenoj pojavi na signal takta se u registar CAR bloka *cpu interfejs* upisuje adresa, flip-flop CRD bloka *cpu interfejs* se postavlja na aktivnu vrednost i prelazi se u korak $step_1$. Postavljanjem flip-flopa CRD na aktivnu vrednost generišu se aktivne vrednosti signala **CRD** i **BUSY** bloka *cpu interfejs*.

U koraku $step_1$ se generiše aktivna vrednost signala **adjLRUCNT** bloka *lru kola*. Pošto je signal saglasnosti **H/M** bloka *tag memorija* aktivan generiše se aktivna vrednost signala **ldCDRRD** bloka *cpu interfejs*. Time se omogućuje upisivanje bajta podatka očitanog iz memorijskog modula DATA bloka *data memorija* u registar CDRRD. Pošto je signal saglasnosti **H/M** aktivan na signal takta se prelazi na korak $step_2$.

U koraku $step_2$ se generišu aktivne vrednosti signala **clCRD** i **CRP** bloka *cpu interfejs*. Signalom **clCRD** flip-flop CRD se postavlja na neaktivnu vrednost. Signalom **CRP** se bajt podatka očitan iz keš memorije ***KEŠ*** prosleđuje iz registra CDRRD bloka *cpu interfejs* u procesor ***CPU*** i procesoru ***CPU*** šalje indikacija da može da produži sa radom. Kada signal **CRD** postane neaktivan i signal **BUSY** bloka *cpu interfejs* postane neaktivan, čime se procesoru ***CPU*** šalje indikacija da keš memorija ***KEŠ*** nije više zauzeta.

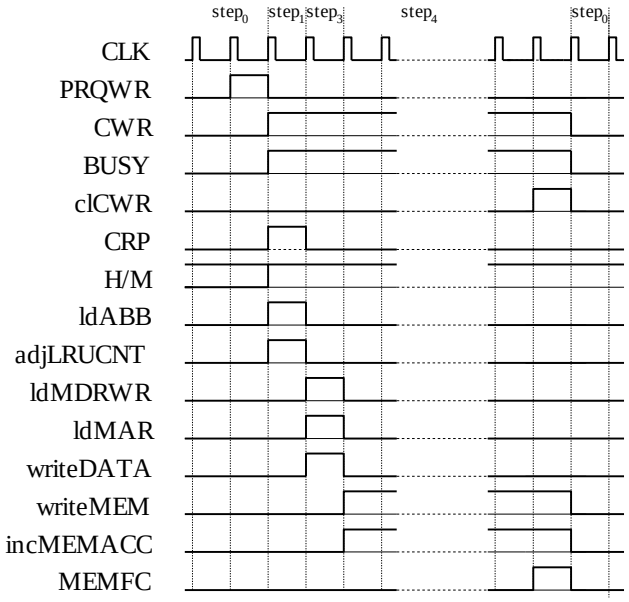
1.2.2.3.2.3.2. Operacija upisa

U slučaju operacije upisa mogu da se jave dva slučaja:

- ima saglasnosti i
- nema saglasnosti.

Vremenski oblici signala za oba slučaja dati su u daljem tekstu.

Vremenski oblici signala za slučaj kada se u keš memoriji ***KEŠ*** izvršava operacija upisa i pri tome ima saglasnosti dati su na slici 57.



Slika 57 Vremenski oblici signala za slučaj operacija upisa i ima saglasnosti

Keš memorija ***KEŠ*** u koraku $step_0$ čeka pojavu aktivne vrednosti signala startovanja operacije čitanja **PRQWR** koji preko bloka *cpu interfejs* dolazi iz procesora ***CPU***. Pri njenoj pojavi na signal takta se u registre CAR i CDRWR bloka *cpu interfejs* upisuju adresa i podatak, respektivno, flip-flop CWR bloka *cpu interfejs* se postavlja na aktivnu vrednost i prelazi se u korak $step_1$. Postavljanjem flip-flopa CWR na aktivnu vrednost generišu se aktivne vrednosti signala **CWR** i **BUSY** bloka *cpu interfejs*.

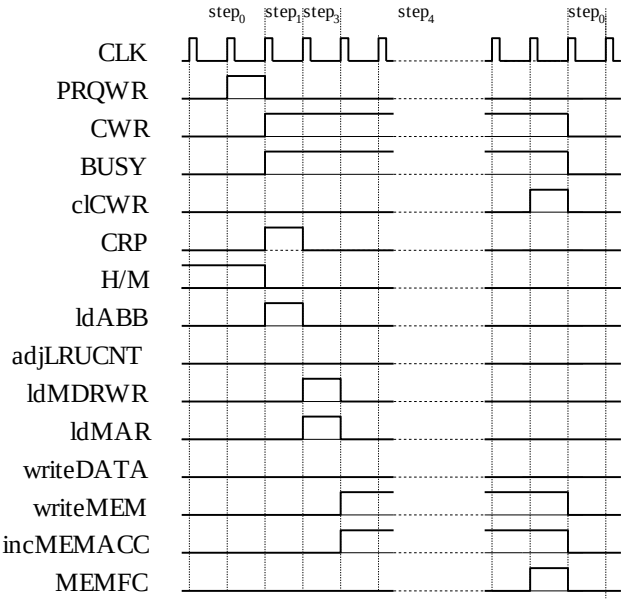
U koraku $step_1$ se generiše aktivne vrednosti signala **ldABB** bloka *tag memorija* i **CRP** bloka *cpu interfejs*. Pošto je signal saglasnosti **H/M** bloka *tag memorija* aktivan generiše se i aktivna vrednost signala **adjLRUCNT** bloka *brojači*. Signalom **CRP** bloka *cpu interfejs* se procesoru ***CPU*** šalje indikacija da može da produži sa radom iako bajt podatka tek treba da se upiše u keš memoriju ***KEŠ*** i memoriju ***MEM***. Međutim, signal **BUSY** bloka *cpu interfejs* ostaje aktivan do kompletnog završetka operacije upisa i sprečava procesor ***CPU*** da startuje novu operaciju čitanja ili upisa. Pošto je signal **CWR** aktivan na signal takta se prelazi na korak $step_3$.

U koraku $step_3$ se generišu aktivne vrednosti signala **ldMDRWR** i **ldMAR** bloka *mem interfejs*. Pošto je signal saglasnosti **H/M** aktivan generiše se i aktivna vrednost signala **writeDATA** bloka *data memorija,* kojom se podatak upisuje u memorijski modul DATA. Iz koraka $step_3$ se na signal takta prelazi na korak $step_4$.

U koraku $step_4$ se ostaje deset perioda takta, jer toliko je potrebno da se izvrši upis podatka u memoriju ***MEM***. U njemu se generišu aktivne vrednosti signala **writeMEM** bloka *mem interfejs* i **incMEMACC** bloka *brojači*, a u desetoj periodi signala takta i signal **MEMFC** bloka *brojači* postaje aktivan. Pri aktivnoj vrednost signala **MEMFC,** generiše se aktivna vrednost signala **clCWR** bloka *cpu interfejs*, pa se flip-flop CWR se postavlja na neaktivnu vrednost. Kada signal **CWR** postane neaktivan i signal **BUSY** postaje neaktivan, čime se procesoru ***CPU*** šalje indikacija da keš memorija ***KEŠ*** nije više zauzeta i da je spremna da

prihvati startovanje nove operacije čitanja ili upisa. Iz koraka $step_4$ se na signal takta prelazi na korak $step_0$.

Vremenski oblici signala za slučaj kada se u keš memoriji ***KEŠ*** izvršava operacija upisa i pri tome nema saglasnosti dati su na slici 58.



Slika 58 Vremenski oblici signala za slučaj operacija upisa i nema saglasnosti

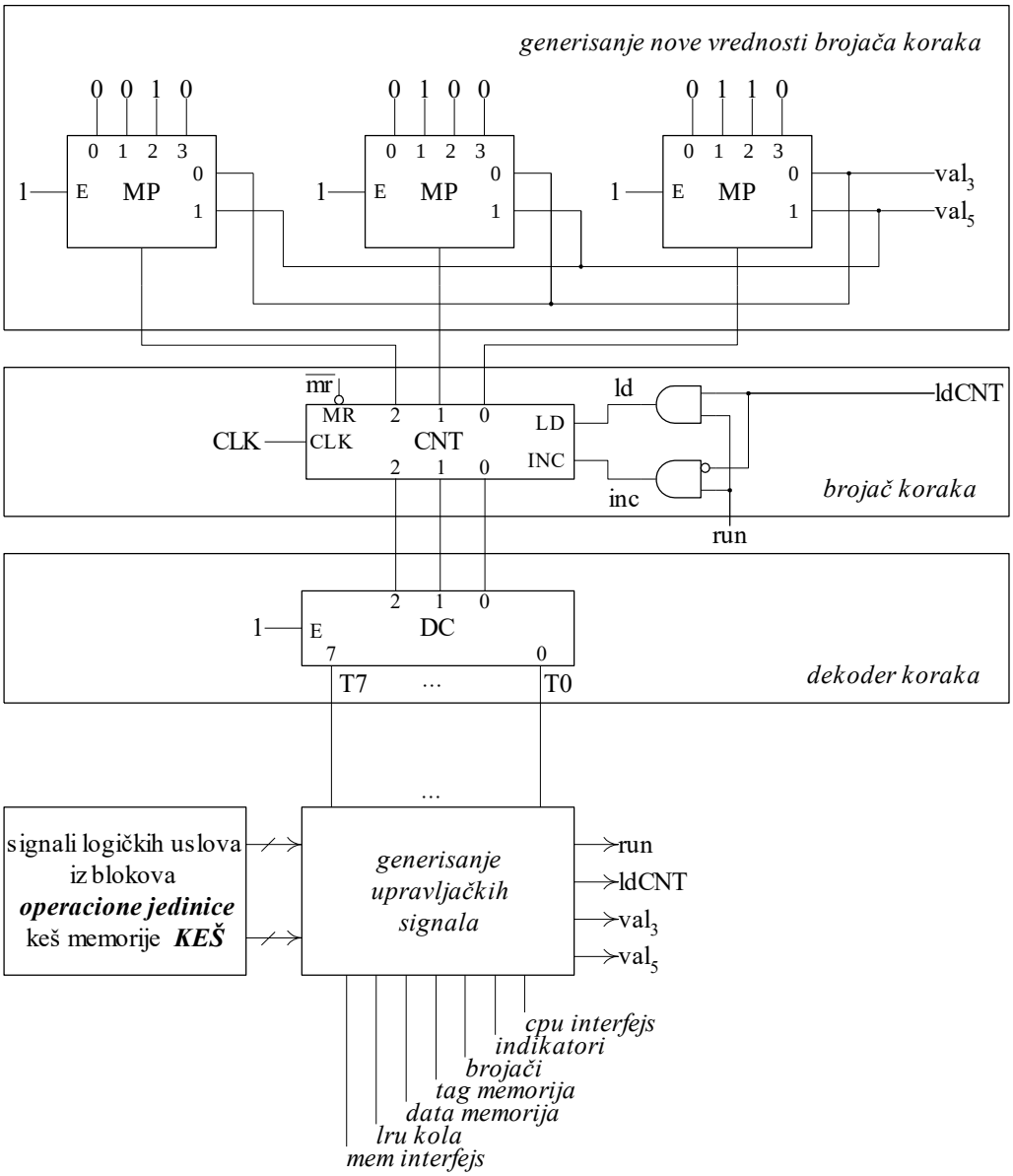
Vremenski oblici signala za slučaj kada se u keš memoriji ***KEŠ*** izvršava operacija upisa i pri tome nema saglasnosti su veoma slični vremenskim oblicima signala za slučaj kada se u keš memoriji ***KEŠ*** izvršava operacija upisa i pri tome ima saglasnosti datim na slici 57. Razlika je samo u tome što su u slučaju kada se u keš memoriji izvršava operacija upisa i pri tome nema saglasnosti upravljački signali **H/M** bloka *tag memorija*, **adjLRUCNT** bloka *brojači* i **writeDATA** bloka *data memorija* na neaktivnoj vrednosti.

1.2.2.3.2.4. Struktura upravljačke jedinice

Struktura ***upravljačke jedinice*** ožičene realizacije je prikazana na slici 59. ***Upravljačka jedinica*** se sastoji iz 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.



Slika 59 Struktura upravljačke jedinice

1.2.2.3.2.4.1. Blok generisanje nove vrednosti brojača koraka

Blok *generisanje nove vrednosti brojača koraka* sadrži multiplekser i služi za generisanje i selekciju vrednosti koju treba upisati u brojač CNT bloka *brojač koraka*. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog generisanja upravljačkih signala. Analizom algoritma generisanja upravljačkih signala (poglavlje 1.2.2.3.2.2) se utvrđuje da su 0, 3 i 5 vrednosti koje treba upisati u brojač. Te vrednosti su ožičene na ulazima 0, 1 i 2 multipleksera, dok se ulaz 3 ne koristi. Selekcija jedne od te tri vrednosti se postiže odgovarajućim vrednostima signala **val_3** i **val_5**. Ako su oba signala neaktivna, kroz multipleksere se propušta vrednost 0, a aktivnom vrednošću jednog od signala **val_3** ili **val_5** kroz multipleksere se propuštaju vrednosti 3 i 5, respektivno.

1.2.2.3.2.4.2. Blok brojač koraka

Blok *brojač koraka* se sastoji od brojača CNT. Brojača 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 1.2.2.3.2.2). 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 vrši se upis nove vrednosti u brojač CNT. Ovim režimom se obezbeđuje odstupanje od sekvencijalnog generisanja upravljačkih signala iz algoritma generisanja upravljačkih signala. Režim skoka se javlja samo onda kada u brojač CNT 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 ne menja se vrednost brojača CNT. Ovaj režim rada se obezbeđuje neaktivnim vrednostima signala **inc** i **ld**. Da bi ovi signali bili neaktivni potrebno je da signal **run** bude neaktivan. Signal **run** je uvek aktivan, sem kada treba obezbediti režim mirovanja, što se dešava kada se čeka:

- pojava signala startovanja operacije čitanja **PRQRD** ili operacije upisa **PRQWR** bloka *cpu interfejs* ili
- pojava signala **MEMFC** bloka *brojači* kojim se označava da je čitanje iz memorije ***MEM*** ili upis u memoriju ***MEM*** obavljeno.

1.2.2.3.2.4.3. Blok dekoder koraka

Blok *dekoder koraka* sadrži dekoder DC. Na ulaze dekodera DC dovede se signali sa izlaza brojača CNT. Dekodovana stanja brojača CNT pojavljuju se kao signali **T0** do **T7** na izlazima dekodera DC. Svakom koraku iz algoritma generisanja upravljačkih signala (poglavlje 1.2.2.3.2.2) dodeljen je jedan od ovih signala i to koraku $step_0$ signal **T0**, koraku $step_1$ signal **T1**, itd.

1.2.2.3.2.4.4. Blok generisanje upravljačkih signala

Blok *generisanje upravljačkih signala* sadrži kombinacione mreže koje pomoću signala **T0** do **T7** koji dolaze iz bloka *dekoder koraka*, signala logičkih uslova koji dolaze iz blokova ***operacione jedinice*** (poglavlje1.2.2.3.1) i saglasno algoritmu generisanja upravljačkih signala (poglavlje 1.2.2.3.2.2) generišu aktivne vrednosti upravljačkih signala. Postupak projektovanja kombinacionih mreža je identičan sa postupkom koji je objašnjen u poglavlju 1.2.1.1.2.4.4.

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

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

1.2.2.3.2.4.4.1. Upravljački signali operacione jedinice

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

- blok *cpu interfejs*,
- blok *indikatori*,
- blok *brojači*,
- blok *tag memorija*,
- blok *data memorija*,
- blok *lru kola* i
- blok *mem interfejs*.

1.2.2.3.2.4.4.1.1. Blok cpu interfejs

Upravljački signali koji odgovaraju ovom bloku ***operacione jedinice*** su:

- **ldCDRRD** — signal paralelnog upisa u registar CDRRD,
- **clCRD** — signal upisa neaktivne vrednosti u flip-flop CRD,
- **clCWR** — signal upisa neaktivne vrednosti u flip-flop CWR i
- **CRP** — signal kompletiranja operacije čitanja ili upisa.

Ovi upravljački signali se generišu na sledeći način:

- **ldCDRRD** = **T1** · **H/M** · **CRD** + **T7** · **CNTBB0**
- **clCRD** = **T2** + **T7** · **CNTBB3**
- **clCWR** = **T4** · **MEMFC** i
- **CRP** = **T1** · **CWR** + **T2** + **T5** · **CNTBB1**.

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

- **H/M** — blok *tag memorija*,
- **CRD** — blok *cpu interfejs*,
- **CWR** — blok *cpu interfejs*,
- **MEMFC** — blok *brojači*,
- **CNTBB0** — blok *brojači*,
- **CNTBB1** — blok *brojači* i
- **CNTBB3** — blok *brojači*.

1.2.2.3.2.4.4.1.2. Blok indikatori

Upravljački signal koji odgovara ovom bloku ***operacione jedinice*** je:

- **writeV** — signal upisa aktivne vrednosti u flip-flop V.

Ovaj upravljački signal se generiše na sledeći način:

- **writeV** = **T7** · **CNTBB3**.

Pri njegovom generisanju koristi se signal logičkog uslova koji dolazi iz bloka ***operacione jedinice*** i to:

- **BBB3** — blok *brojači*.

1.2.2.3.2.4.4.1.3. Blok brojači

Upravljački signali koji odgovaraju ovom bloku ***operacione jedinice*** su:

- **incABB** — signal inkrementiranja sadržaja brojača ABB,
- **incCNTBB** — signal inkrementiranja sadržaja brojača CNTBB,

- **incMEMACC** — signal inkrementiranja sadržaja brojača MEMACC i
- **ldABB** — signal paralelnog upisa u brojač ABB.

Ovi upravljački signali se generišu na sledeći način:

- **incABB** = **T7**,
- **incCNTBB** = **T7**,
- **incMEMACC** = **T4** + **T6** i
- **ldABB** = **T1** · ($\overline{\textbf{H/M}}$ · **CRD** + **CWR**).

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

- $\overline{\textbf{H/M}}$ — blok *tag memorija*,
- **CRD** — blok *cpu interfejs* i
- **CWR** — blok *cpu interfejs*.

1.2.2.3.2.4.4.1.4. Blok tag memorija

Upravljački signal koji odgovara ovom bloku ***operacione jedinice*** je:

- **writeTAG** — signal upisa u memorijski modul TAG.

Ovaj upravljački signali se generiše na sledeći način:

- **writeTAG** = **T7** · **CNTBB3**.

Pri njegovom generisanju koristi se signal logičkog uslova koji dolazi iz bloka ***operacione jedinice*** i to:

- **CNTBB3** — blok *brojači*.

1.2.2.3.2.4.4.1.5. Blok data memorija

Upravljački signali koji odgovaraju ovom bloku ***operacione jedinice*** su:

- **mxDIDATA** — signal selekcije kroz multiplekser,
- **mxADATA** — signal selekcije kroz multiplekser
- **writeDATA** — signal upisa u memorijski modul DATA i
- **mxDATA** — signal selekcije kroz multiplekser.

Ovi upravljački signali se generišu na sledeći način:

- **mxDIDATA** = **T7**,
- **mxADATA** = **T7**,
- **incMEMACC** = **T3** · **H/M** + **T7** i
- **mxDATA** = **T7** · **CNTBB0**.

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

- **H/M** — blok *tag memorija* i
- **CNTBB0** — blok *brojači*.

1.2.2.3.2.4.4.1.6. Blok lru kola

Upravljački signal koji odgovara ovom bloku ***operacione jedinice*** je:

- **adjLRUCNT** — signal ažuriranja sadržaja LRU brojača.

Ovaj upravljački signali se generiše na sledeći način:

- **adjLRUCNT** = **T1** · (**CRD** + **CWR**) · **H/M** + **T7** · **CNTBB3**.

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

- **H/M** — blok *tag memorija*,
- **CRD —** blok *cpu interfejs*,
- **CWR** — blok *cpu interfejs*, i
- **CNTBB3** — blok brojači.

1.2.2.3.2.4.4.1.7. Blok mem interfejs

Upravljački signali koji odgovaraju ovom bloku ***operacione jedinice*** su:

- **ldMAR** — signal paralelnog upisa u registar MAR,
- **ldMDRWR** — signal paralelnog upisa u registar MDRWR,
- **ldMDRRD** — signal paralelnog upisa u registar MDRRD i
- **writeMEM** — signal upisa u memoriju ***MEM***.

Ovi upravljački signali se generišu na sledeći način:

- **ldMAR** = **T3** + **T5**,
- **ldMDRWR** = **T3**,
- **ldMDRRD** = **T6** · **MEMFC** i
- **writeMEM** = **T4**.

Pri njihovom generisanju koristi se signal logičkog uslova koji dolazi iz bloka ***operacione jedinice*** i to:

- **MEMFC** — blok *brojači*.

1.2.2.3.2.4.4.2. Upravljački signali upravljačke jedinice.

Upravljački signali upravljačke jedinice su:

- **ldCNT** — signal čijom se aktivnom vrednošću, pod uslovom da je signal **run** aktivan, upisuje 0, 3 ili 5 u brojač CNT, a neaktivnom vrednošću, pod uslovom da je signal **run** aktivan, inkrementira tekuća vrednosti brojača CNT,
- **run** — signal čijom se aktivnom vrednošću omogućuje, u zavisnosti od vrednosti signala **ldCNT,** ili upis nove vrednosti ili inkrementiranje vrednosti brojača CNT, a neaktivnom vrednošću, bez obzira na vrednost signala **ldCNT**, onemogućava promena vrednosti brojača CNT i
- $\mathbf{val_3}$ i $\mathbf{val_5}$ — signali koji obezbeđuju generisanje vrednosti 0, 3 i 5 za upis u brojač CNT.

Ovi upravljački signali se generišu na sledeći način:

- $\mathbf{ldCNT} = \mathbf{T1} \cdot (\mathbf{CWR} + \overline{\mathbf{H/M}}) + \mathbf{T2} + \mathbf{T4} \cdot \mathbf{MEMFC} + \mathbf{T7}$,
- $\mathbf{run} = \mathbf{T0} \cdot (\mathbf{PRQRD} + \mathbf{PRQWR}) + \mathbf{T1} + \mathbf{T2} + \mathbf{T3} + \mathbf{T4} \cdot \mathbf{MEMFC} + \mathbf{T5} + \mathbf{T6} \cdot \mathbf{MEMFC} + \mathbf{T7}$,
- $\mathbf{val_3} = \mathbf{T1} \cdot \mathbf{CWR}$ i
- $\mathbf{val_5} = \mathbf{T1} \cdot \overline{\mathbf{CWR}} \cdot \overline{\mathbf{H/M}} + \mathbf{T7} \cdot \overline{\mathbf{CNTBB3}}$.

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

- **H/M** — blok *tag memorija*,
- **CWR** — blok *cpu interfejs*,
- **MEMFC** — blok *brojači*,
- **CNTBB3** — blok *brojači*,
- **PRQRD** — blok *cpu interfejs* i

- **PRQWR** — blok *cpu interfejs*.

1.2.3. KEŠ MEMORIJA SA SET-ASOCIJATIVNIM PRESLIKAVANJEM

Keš memorija koja se razmatra realizovana je tehnikom set-asocijativnog preslikavanja na nivou bloka veličine 4 bajta. U ovoj tehnici preslikavanja **k**-ti blok bilo koje grupe operativne memorije može da se smesti u bilo koji ulaz unutar **k**-tog seta keš memorije, pa je stoga unutar seta neophodno koristiti neki od algoritama zamene. U usvojenoj realizaciji koristi se FIFO algoritam zamene. Za ažuriranja operativne memorije koristi se tehnika *vrati-nazad*. Radi smanjivanja čekanja procesora koriste se tehnike *baferovanje, prosleđivanje* i *rani start procesora*. Tehnike *baferovanje* i *prosleđivanje* se koriste samo kod operacije čitanja. Kada se u slučaju operacije čitanja otkrije da nema saglasnosti blok podataka koji je odabran za zamenu i koji je modifikovan se najpre prebacuje iz keš memorije u bafer podataka, zatim se dovlači zahtevani blok podataka iz operativne memorije u keš memorije i na kraju se vrši prebacivanje bloka podataka iz bafera podataka u operativnu memoriju. Kod dovlačenja zahtevanog bloka podataka iz operativne memorije u keš memorije najpre se dovlači zahtevani bajt unutar bloka, pa tek onda preostali bajtovi bloka. Pri tome se dovučeni zahtevani bajt bloka odmah prosleđuje i procesoru. Tehnika *rani start procesora* se koristi kod svih operacija, ali na različite načine. Kod operacije čitanja keš memorija u trunutku slanja bajta podatka procesoru signalizira da procesor može da produži sa radom. Pri tome je keš memorija kompletno završila operaciju čitanja ukoliko je otkrivena saglasnost. Međutim, ukoliko nije otkrivena saglasnost, keš memorija nije još uvek kompletno završila operaciju čitanja. Tada treba da se dovuku i preostali bajtovi bloka iz operativne memorije u keš memoriju, a u slučaju da je blok odabran za zamenu modifikovan i da se bajtovi datog bloka prebace iz bafera bloka u operativnu memoriju. Kod preostalih operacija keš memorija signalizira da procesor može da produži sa radom čim od procesora prihvati informacije neophodne za realizaciju odgovarajuće operacije iako nije još uvek kompletno završila operaciju. Kao sastavni deo tehnike *rani start procesora* keš memorija mora i da signalizira procesoru da li je operacija završena, da bi se sprečilo startovanje nove operacije pre nego što je prethodno startovana operacija završena.

Keš memorija ima 4 seta sa po 2 ulaza po setu u kojima se čuva 8 blokova operativne memorije. S obzirom da je blok veličine 4 bajta, kapacitet dela keš memorije u kome se čuva sadržaj je 32 bajta, a adresibilna jedinica je jedan bajt.

Operativna memorija je kapaciteta 64 kB, a adresibilna jedinica je jedan bajt. Stoga se operativna memorija može posmatrati kao da je organizovana u 2^{12} grupa, od kojih svaka sadrži 2^2 blokova veličine 2^2 bajta. Adresa operativne memorije dužine 16 bita može se podeliti i označiti na sledeći način:

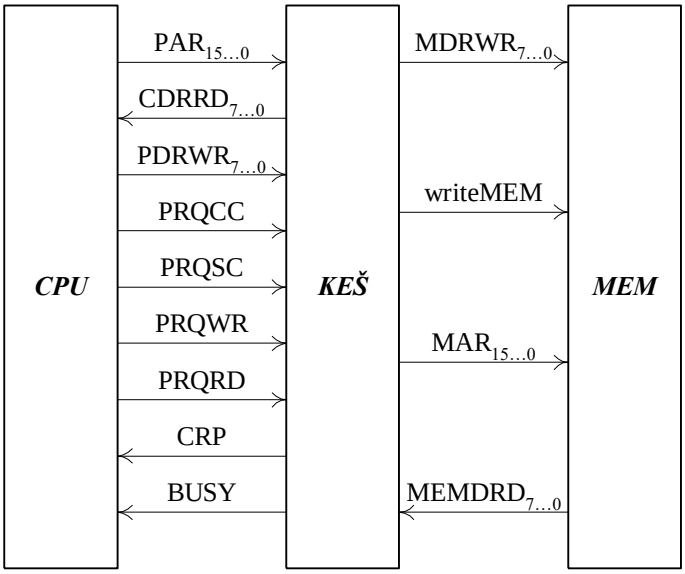
- najviših 12 bitova označavaju broj grupe,
- srednja 2 bita označavaju broj seta i
- najniža 2 bita označavaju adresu bajta u bloku.

Keš memorija je deo sistema (slika 60) koji se sastoji iz:

- procesora ***CPU***,
- memorije ***MEM*** i
- keš memorije ***KEŠ***.

Uzeto je da su sve sekvencijalne mreže sistema sinhrone i da ceo sistem radi sinhrono sa zajedničkim signalom takta **CLK**.

U ovom poglavlju se daju detalji realizacije procesora ***CPU*** i memorije ***MEM*** relevantni za rad keš memorije ***KEŠ*** i kompletna hardverska realizacija keš memorije ***KEŠ***.



Slika 60 Struktura sistema

1.2.3.1. PROCESOR CPU

Procesor ***CPU*** se obraća keš memoriji ***KEŠ*** onda kada treba očitati podatak iz keš memorije ***KEŠ***, upisati podatak u keš memoriju ***KEŠ***, vratiti iz keš memorije ***KEŠ*** u memoriju ***MEM*** određeni blok podataka ukoliko je blok modifikovan ili vratiti iz keš memorije ***KEŠ*** u memoriju ***MEM*** sve blokove podataka koji su modifikovani. Ova četiri obraćanja procesora ***CPU*** keš memoriji ***KEŠ*** se u daljem tekstu nazivaju operacija čitanja, operacija upisa, operacija selektivnog vraćanja i operacija kompletnog vraćanja.

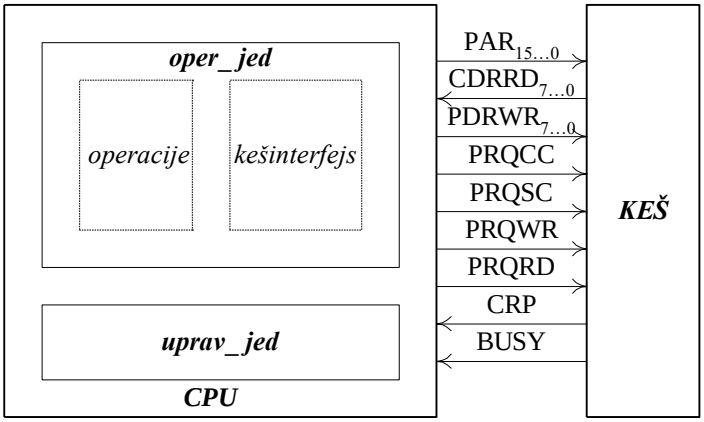
Operacije čitanja ili upisa se koriste kad god se generiše adresa memorije ***MEM*** sa koje treba očitati ili instrukciju ili operand ili na kojoj treba upisati rezultat.

Operacija selektivnog vraćanja se koristi onda kada sve modifikovane blokove iz određenog opsega adresa memorije ***MEM*** treba vratiti iz keš memorije ***KEŠ*** u memoriju ***MEM***. Ukoliko taj opseg adresa sadrži *n* blokova, procesor ***CPU*** generiše *n* operacija selektivnog vraćanja. Svaka operacija selektivnog vraćanja specificira i adresu jednog od bajtova bloka. Kod ove operacije keš memorija ***KEŠ*** vrši proveru da li se podatak sa zadate adrese nalazi u odgovarajućem bloku keš memorije ***KEŠ***. U slučaju da se nalazi, vrši se provera da li je blok modifikovan. Ukoliko jeste, blok se vraća iz keš memorije ***KEŠ*** u memoriju ***MEM*** i operacija završava. Ukoliko nije, operacija se odmah završava. U slučaju da se blok ne nalazi u keš memoriji ***KEŠ***, operacija se završava.

Operacija kompletnog vraćanja se koristi onda kada treba sve modifikovane blokove keš memorije ***KEŠ*** vratiti iz keš memorije ***KEŠ*** u memoriju ***MEM***. Kod ove operacije nema specificiranja adrese, jer se vrši provera redom za sve blokove keš memorije ***KEŠ*** da li su modifikovani. Onaj blok za koji se utvrdi da je modifikovan vraća se iz keš memorije ***KEŠ*** u memoriju ***MEM***, pa se prelazi na proveru sledećeg bloka. Onaj blok za koji se utvrdi da nije modifikovan ne dira se, već se odmah prelazi na proveru sledećeg bloka. Operacija kompletnog vraćanja se završava pošto se opisani postupak realizuje za sve blokove keš memorije ***KEŠ***.

Operacije selektivnog i kompletnog vraćanja se koriste kod softverskog održavanja koherencije operativne memorije i keš memorije i to u slučaju kada se za ažuriranje sadržaja operativne memorije koristi tehnika *vrati nazad*. Operacije selektivnog vraćanja se koriste kada to treba uraditi samo za određeni opseg adresa operativne memorije, a operacija kompletnog vraćanja se koristi kada to treba uraditi za kompletan opseg adresa operativne memorije.

Signali koji se razmenjuju između procesora ***CPU*** i keš memorije ***KEŠ*** prilikom izvršavanja ove četiri operacije prikazani su na slici 61.



Slika 61 Organizacija procesora ***CPU***

Kod operacije čitanja procesor ***CPU*** šalje keš memoriji ***KEŠ*** 16-bitnu adresu po linijama **PAR$_{15...0}$** i generiše aktivnu vrednost signala **PRQRD** trajanja jedna perioda signala takta. Keš memorija ***KEŠ*** vraća očitani 8-bitni podatak po linijama **CDRRD$_{7...0}$**. Keš memorija ***KEŠ*** šalje procesoru ***CPU*** i signale **CRP** i **BUSY**. Aktivnom vrednošću signala **CRP** trajanja jedna perioda signala takta keš memorija ***KEŠ*** signalizira procesoru ***CPU*** da je na linijama **CDRRD$_{7...0}$** važeći podatak i da procesor ***CPU*** može da produži sa radom. Aktivnom i neaktivnom vrednošću signal **BUSY** keš memorija ***KEŠ*** signalizira procesoru ***CPU*** da li je zauzeta ili ne, respektivno. Kada procesor ***CPU*** aktivnom vrednošću signala **PRQRD** startuje u keš memoriji ***KEŠ*** operaciju čitanja, keš memorija ***KEŠ*** upisuje adresu sa linija **PAR$_{7...0}$** u prihvatni registar adrese, generiše aktivnu vrednost signala **CRP** i signal **BUSY** postavlja na aktivnu vrednost. U slučaju da u keš memoriji ***KEŠ*** ima saglasnosti očitani podatak se šalje u procesor ***CPU***, generiše aktivna vrednost signala **CRP** i signal **BUSY** postavlja na neaktivnu vrednost. Time se procesoru ***CPU*** omogućava da produži sa radom i eventualno startuje novu operaciju u keš memoriji ***KEŠ***. U slučaju da u keš memoriji ***KEŠ*** nema saglasnosti, blok podataka odabran u keš memoriji ***KEŠ*** za zamenu se u slučaju da je modifikovan prebacuje iz keš memorije ***KEŠ*** u bafer bloka, iz memorije ***MEM*** se u keš memoriju ***KEŠ*** dovlači najpre bajt podatka sa generisane adrese, očitani podatak odmah šalje u procesor ***CPU*** i generiše aktivna vrednost signala **CRP**, čime se procesoru dozvoljava da produži sa radom. Međutim, signal **BUSY** ostaje aktivan sve dok je keš memorija ***KEŠ*** zauzeta dovlačenjem preostalih bajtova bloka, a ukoliko je blok odabran za zamenu modifikovan i prebacivanjem bloka podataka iz bafera bloka u memoriju ***MEM***, čime se procesoru ***CPU*** onemogućava da startuje novu operaciju u keš memoriji ***KEŠ*** pre kompletiranja prenosa svih bajtova bloka iz memorije ***MEM*** u keš memoriju ***KEŠ.***

Kod operacije upisa procesor ***CPU*** šalje keš memoriji ***KEŠ*** 16-bitnu adresu po linijama **PAR$_{15...0}$**, 8-bitni podatak za upis po linijama **PDRWR$_{7...0}$** i generiše aktivnu vrednost signala **PRQWR** trajanja jedna perioda signala takta. Keš memorija ***KEŠ*** šalje procesoru ***CPU***

signale **CRP** i **BUSY**. Aktivnom vrednošću signala **CRP** trajanja jedna perioda signala takta keš memorija ***KEŠ*** signalizira procesoru ***CPU*** da procesor ***CPU*** može da produži sa radom. Aktivnom i neaktivnom vrednošću signal **BUSY** keš memorija ***KEŠ*** signalizira procesoru ***CPU*** da li je zauzeta ili ne, respektivno. Kada procesor ***CPU*** aktivnom vrednošću signala **PRQWR** startuje u keš memoriji ***KEŠ*** operaciju upisa, keš memorija ***KEŠ*** upisuje adresu sa linija $\mathbf{PAR_{7...0}}$ u prihvatni registar adrese i podatak sa linija $\mathbf{PDRWR_{7...0}}$ u prihvatni registar podatka, generiše aktivnu vrednost signala **CRP** i signal **BUSY** postavlja na aktivnu vrednost. Time se procesoru ***CPU*** omogućava da produži sa radom. Međutim, signal **BUSY** je aktivan sve dok je keš memorija ***KEŠ*** zauzeta realizovanjem operacije upisa. U slučaju da u keš memoriji ***KEŠ*** ima saglasnosti, signal **BUSY** je aktivan dok je keš memorija ***KEŠ*** zauzeta upisom podatka iz prihvatnog registra podatka u keš memoriju ***KEŠ***. U slučaju da u keš memoriji ***KEŠ*** nema saglasnosti, signal **BUSY** je aktivan sve dok je keš memorija ***KEŠ*** zauzeta prebacivanjem bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***, i to samo ukoliko je blok odabran za zamenu modifikovan, upisom podatka iz prihvatnog registra podatka u keš memoriju ***KEŠ*** i dovlačenjem preostalih bajtova bloka iz memoriju ***MEM*** u keš memoriji ***KEŠ***. Time se onemogućava procesoru ***CPU*** da startuje novu operaciju u keš memoriji ***KEŠ*** pre kompletiranja operacije upisa**.**

Kod operacije selektivnog vraćanja procesor ***CPU*** šalje keš memoriji ***KEŠ*** 16-bitnu adresu po linijama $\mathbf{PAR_{15...0}}$ i generiše aktivnu vrednost signala **PRQSC** trajanja jedna perioda signala takta. Keš memorija ***KEŠ*** šalje procesoru ***CPU*** signale **CRP** i **BUSY**. Aktivnom vrednošću signala **CRP** trajanja jedna perioda signala takta keš memorija ***KEŠ*** signalizira procesoru ***CPU*** da procesor ***CPU*** može da produži sa radom. Aktivnom i neaktivnom vrednošću signal **BUSY** keš memorija ***KEŠ*** signalizira procesoru ***CPU*** da li je zauzeta ili ne, respektivno. Kada procesor ***CPU*** aktivnom vrednošću signala **PRQSC** startuje u keš memoriji ***KEŠ*** operaciju selektivnog vraćanja, keš memorija ***KEŠ*** upisuje adresu sa linija $\mathbf{PAR_{7...0}}$ u prihvatni registar adrese, generiše aktivnu vrednost signala **CRP** i signal **BUSY** postavlja na aktivnu vrednost. Time se procesoru ***CPU*** omogućava da produži sa radom. Međutim, signal **BUSY** je aktivan sve dok je keš memorija ***KEŠ*** zauzeta realizovanjem operacije selektivnog vraćanja. U slučaju da u keš memoriji ***KEŠ*** ima saglasnosti i blok je modifikovan, signal **BUSY** postaje neaktivan tek kada keš memorija ***KEŠ*** završi prebacivanje bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***. U slučaju da u keš memoriji ***KEŠ*** ima saglasnosti i blok nije modifikovan, signal **BUSY** postaje odmah neaktivan. U slučaju da u keš memoriji ***KEŠ*** nema saglasnosti, signal **BUSY** postaje odmah neaktivan. Time se onemogućava procesoru ***CPU*** da startuje novu operaciju u keš memoriji ***KEŠ*** pre kompletiranja operacije selektivno vračanje**.**

Kod operacije kompletnog vraćanja procesor ***CPU*** generiše aktivnu vrednost signala **PRQCC** trajanja jedna perioda signala takta. Keš memorija ***KEŠ*** šalje procesoru ***CPU*** signale **CRP** i **BUSY**. Aktivnom vrednošću signala **CRP** trajanja jedna perioda signala takta keš memorija ***KEŠ*** signalizira procesoru ***CPU*** da procesor ***CPU*** može da produži sa radom. Aktivnom i neaktivnom vrednošću signal **BUSY** keš memorija ***KEŠ*** signalizira procesoru ***CPU*** da li je zauzeta ili ne, respektivno. Kada procesor ***CPU*** aktivnom vrednošću signala **PRQCC** startuje u keš memoriji ***KEŠ*** operaciju kompletnog vraćanja, keš memorija ***KEŠ*** generiše aktivnu vrednost signala **CRP** i signal **BUSY** postavlja na aktivnu vrednost. Time se procesoru ***CPU*** omogućava da produži sa radom. Međutim, signal **BUSY** je aktivan sve dok je keš memorija ***KEŠ*** zauzeta realizovanjem operacije kompletnog vraćanja. Signal **BUSY** postaje neaktivan tek kada keš memorija ***KEŠ*** završi prebacivanje iz keš memorije ***KEŠ*** u memoriju ***MEM*** svih blokova keš memorije ***KEŠ*** koji su modifikovani. podataka iz. Time se procesoru ***CPU*** onemogućava da startuje novu operaciju u keš memoriji ***KEŠ*** pre kompletiranja operacije kompletno vraćanje**.**

Procesor ***CPU*** se sastoji iz operacione jedinice ***oper_ jed*** i upravljačke jedinice ***uprav_ jed*** (slika 61). Operaciona jedinica ***oper_jed*** je kompozicija kombinacionih i sekvencijalnih pre mreža koje služe za pamćenje binarnih reči, izvršavanje mikrooperacija i generisanje signala logičkih uslova. Upravljačka jedinica ***uprav_jed*** je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za generisanje upravljačkih signala prema algoritmu generisanja upravljačkih signala operacija procesora ***CPU*** i signala logičkih uslova. Struktura i opis operacione jedinice i upravljačke jedinice se daju u daljem tekstu.

1.2.3.1.1. Operaciona jedinica

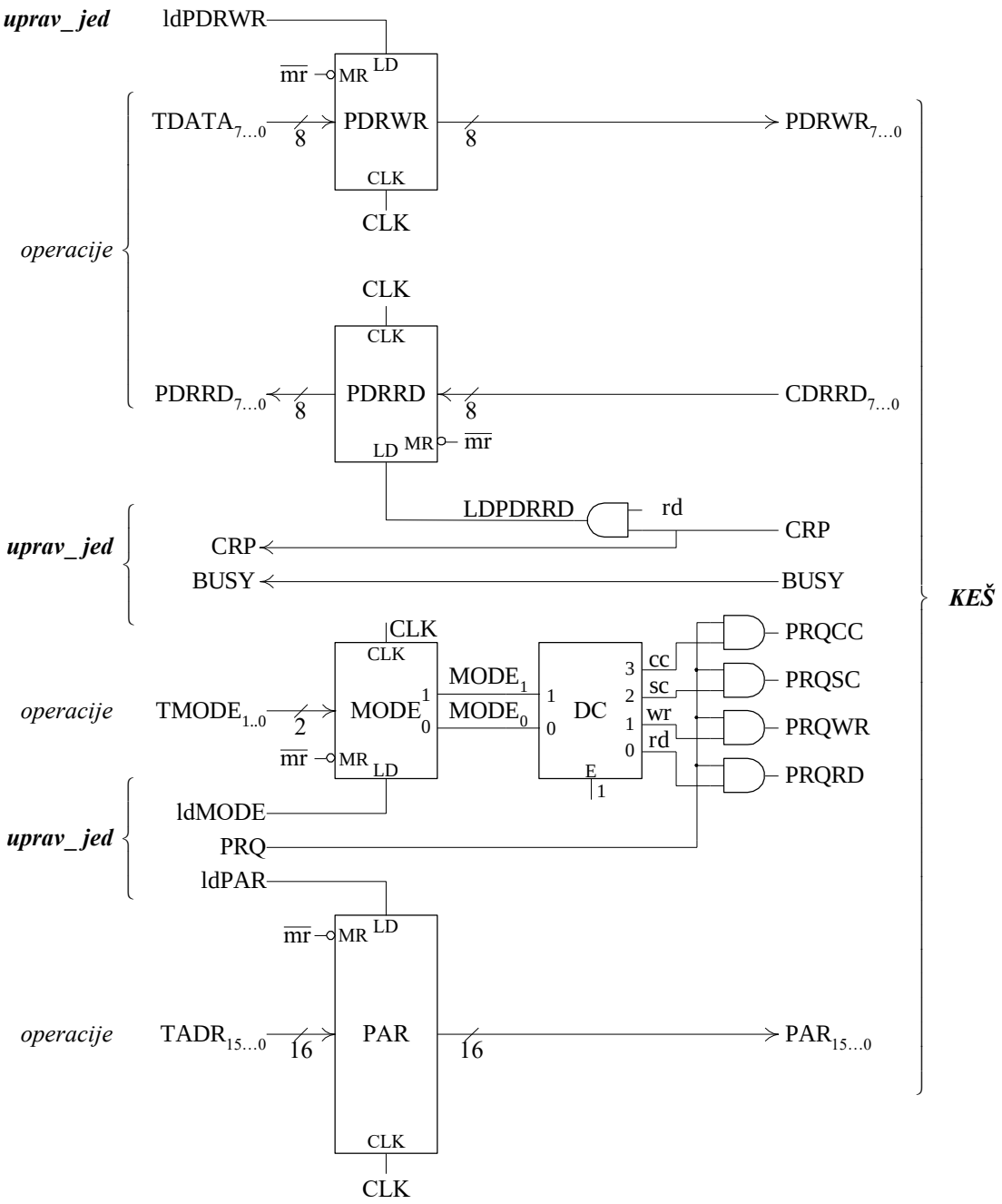
Operaciona jedinica ***oper_ jed*** (slika 61) se sastoji od sledećih blokova:

- blok *kešinterfejs* i
- blok *operacije*.

Blok *keš interfejs* služi za povezivanje procesora ***CPU*** i keš memorije ***KEŠ***. Blok *operacije* služi za generisanje operacija čitanja, upisa, selektivnog vraćanja i kompletnog vraćanja. Struktura i opis blokova operacione jedinice ***oper_ jed*** se daju u daljem tekstu.

1.2.3.1.1.1. Blok kešinterfejs

Blok *kešinterfejs* (slika 62) sadrži registre PAR, PDRWR, PDRRD i MODE, dekoder DC i 4 logička I kola.



Slika 62 Blok kešinterfejs

Registar PAR (Processor Address Register) služi za čuvanje ili adrese lokacije memorije ***MEM*** sa koje treba očitati sadržaj u slučaju operacije čitanja ili adrese lokacije memorije ***MEM*** u koju treba upisati sadržaj u slučaju operacije upisa ili adrese lokacije memorije ***MEM*** koja pripada bloku koji treba, ukoliko je modifikovan, vratiti iz keš memorije ***KEŠ*** u memoriju ***MEM*** u slučaju operacije selektivnog vraćanja. Izlazne linije ovog registra se vode kao 16 adresnih linija $\mathbf{PAR_{15...0}}$ u keš memoriju ***KEŠ***. Pretpostavlja se da je procesor ***CPU*** pre obraćanja keš memoriji ***KEŠ*** generisanjem aktivne vrednosti signala **ldPAR** trajanja jedna perioda signala takta na signal takta izvršio upis adrese sa linija $\mathbf{TADR_{15...0}}$ u registar PAR.

Registar PDRWR (Processor Data Register for WRite) služi za čuvanje sadržaja koji treba upisati kod operacije upisa. Izlazne linije ovog registra se vode kao 8 linija podataka $\mathbf{PDRWR_{7...0}}$ u keš memoriju ***KEŠ***. Pretpostavlja se da je procesor ***CPU*** pre obraćanja keš

memoriji ***KEŠ*** već upisao sadržaj u registar PDRWR, tako što je generisao aktivnu vrednost signala **ldPDRWR** trajanja jedna perioda signala takta i na signal takta izvršio upis sadržaja sa linija $\mathbf{TDATA_{7...0}}$.

Registar PDRRD (Processor Data Register for ReaD) služi za čuvanje sadržaja koji je očitan kod operacije čitanja. Očitani podatak se iz keš memorije ***KEŠ*** vodi po linijama $\mathbf{CDRRD_{7...0}}$ na ulaze registra PDRRD. U slučaju operacije čitanja signal **rd** je aktivan, pa se aktivnom vrednošću signala **CRP** trajanja jedna perioda signala takta na signal takta izvrši upis očitanog sadržaja sa linija $\mathbf{CDRRD_{7...0}}$ u registar PDRRD.

Registar MODE (operation MODE) služi za čuvanje binarne vrednosti jedne od četiri operacije koje keš memorija ***KEŠ*** može da realizuje. Pretpostavlja se da je procesor ***CPU*** pre obraćanja keš memoriji ***KEŠ*** već upisao binarnu vrednost operacije u registar MODE, tako što je generisao aktivnu vrednost signala **ldMODE** trajanja jedna perioda signala takta i najkasnije na signal takta kojim se signal **PRQ** (Processor ReQuest) postavlja na aktivnu vrednost izvršio upis sadržaja sa linija $\mathbf{TMODE_{1...0}}$ u registar MODE. Signali $\mathbf{MODE_{1...0}}$ sa izlaza registra MODE se vode na ulaze dekodera DC.

Dekoder DC i četiri logička I kola služe za generisanje signala **PRQRD**, **PRQWR**, **PRQSC** i **PRQCC**. Dekoder DC na svojim izlazima 0, 1, 2 i 3 daje aktivnu vrednost jednog od signala **rd**, **wr**, **sc** i **cc** u zavisnosti od binarne vrednosti signala $\mathbf{MODE_{1...0}}$ sa ulaza. Pri aktivnoj vrednosti signala **PRQ** trajanja jedna perioda signala takta jedan od signala **PRQRD**, **PRQWR**, **PRQSC** i **PRQCC** postaje aktivan u zavisnosti od toga koji je od signala **rd**, **wr**, **sc** i **cc** aktivan. Signali **PRQRD**, **PRQWR**, **PRQSC** i **PRQCC** su neaktivni pri neaktivnoj vrednosti signala **PRQ**.

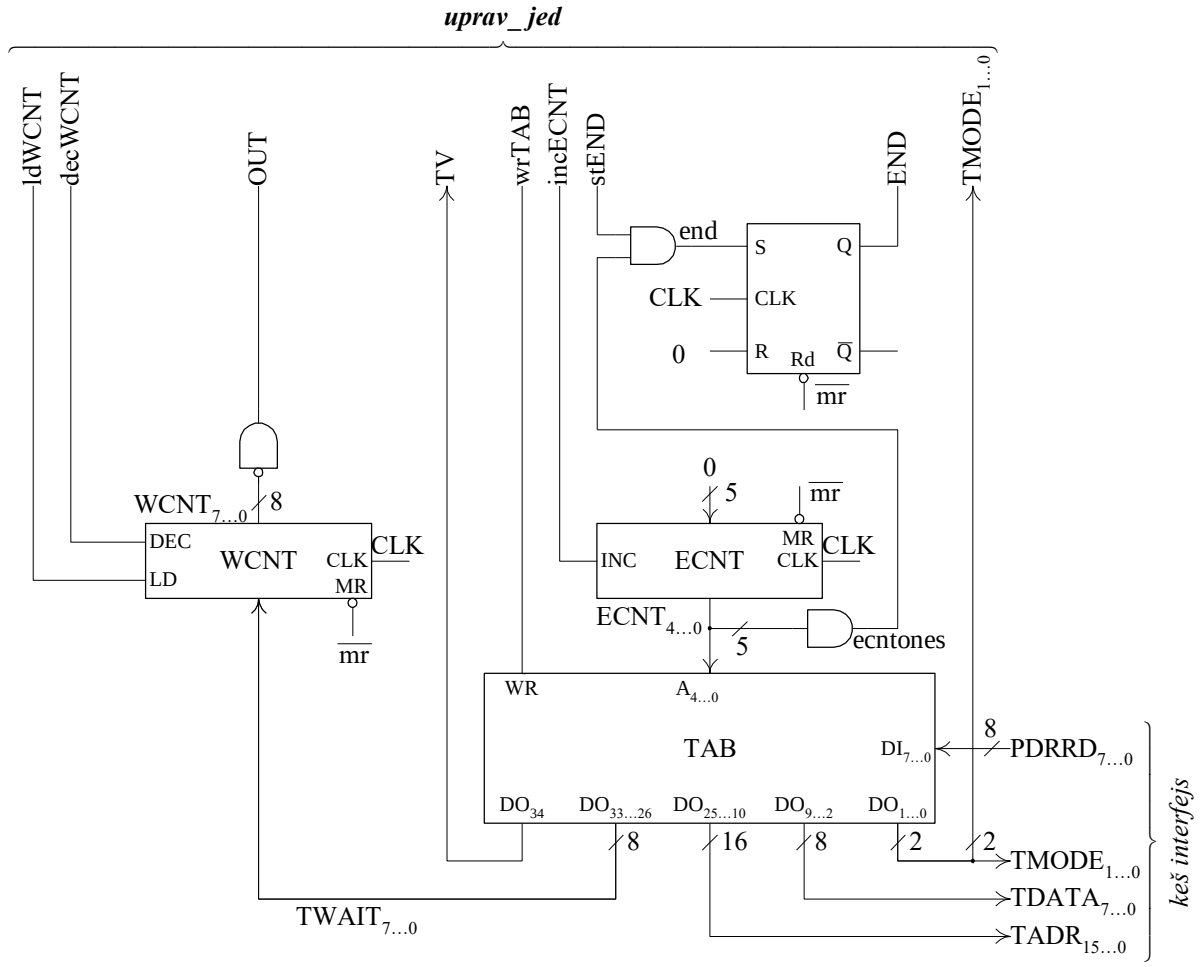
Upravljački signali **PRQRD** (Processor ReQuest for ReaD), **PRQWR** (Processor ReQuest for WRite), **PRQSC** (Processor ReQuest for Selective Clear) i **PRQCC** (Processor ReQuest for Complete Clear) imaju identičnu funkciju kao upravljački signali sa identičnim nazivima u odeljku 1.2.1.1.1.1.

Upravljački signal **CRP** (Cache RePly) se koristi da keš memorija ***KEŠ*** aktivnom vrednošću ovog signala trajanja jedna perioda signala takta signalizira procesoru ***CPU*** da može da produži sa radom iako operacija čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja nije završena. U slučaju operacije čitanja aktivnom vrednošću signala **CRP** se signalizira i da se na linijama $\mathbf{CDRRD_{7...0}}$ nalazi očitani podatak i da aktivnom vrednošću signala **CRP** treba na signal takta sadržaj sa linija $\mathbf{CDRRD_{7...0}}$ upisati u registar PDRRD.

Upravljački signal **BUSY** se koristi da keš memorija ***KEŠ*** aktivnom vrednošću ovog signala signalizira procesoru ***CPU*** da je još uvek zauzeta jer sve aktivnosti u keš memoriji ***KEŠ*** vezane za operaciju čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja nisu završene i da procesor ***CPU*** ne sme da startuje novu operaciju čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja.

1.2.3.1.1.2. Blok operacije

Blok *operacije* (slika 63) sadrži brojače ECNT i WCNT, flip-flop END i memoriju TAB.



Slika 63 Blok operacije

Brojači ECNT (Entry CouNTer) i WCNT (Wait CouNTer) i flip-flop END (generation of operations ENDed) imaju identičnu funkciju kao i registri i flip-flop sa identičnim nazivima u odeljku 1.2.1.1.1.2.

Memorija TAB ima 32 ulaza, pa najviše 32 operacije keš memorije ***KEŠ*** mogu da budu generisane. Jedan ulaz sadrži informacije na osnovu kojih se generiše jedna operacija keš memorije ***KEŠ*** (slika 64).

Slika 64 Struktura jednog ulaza u memoriji TAB

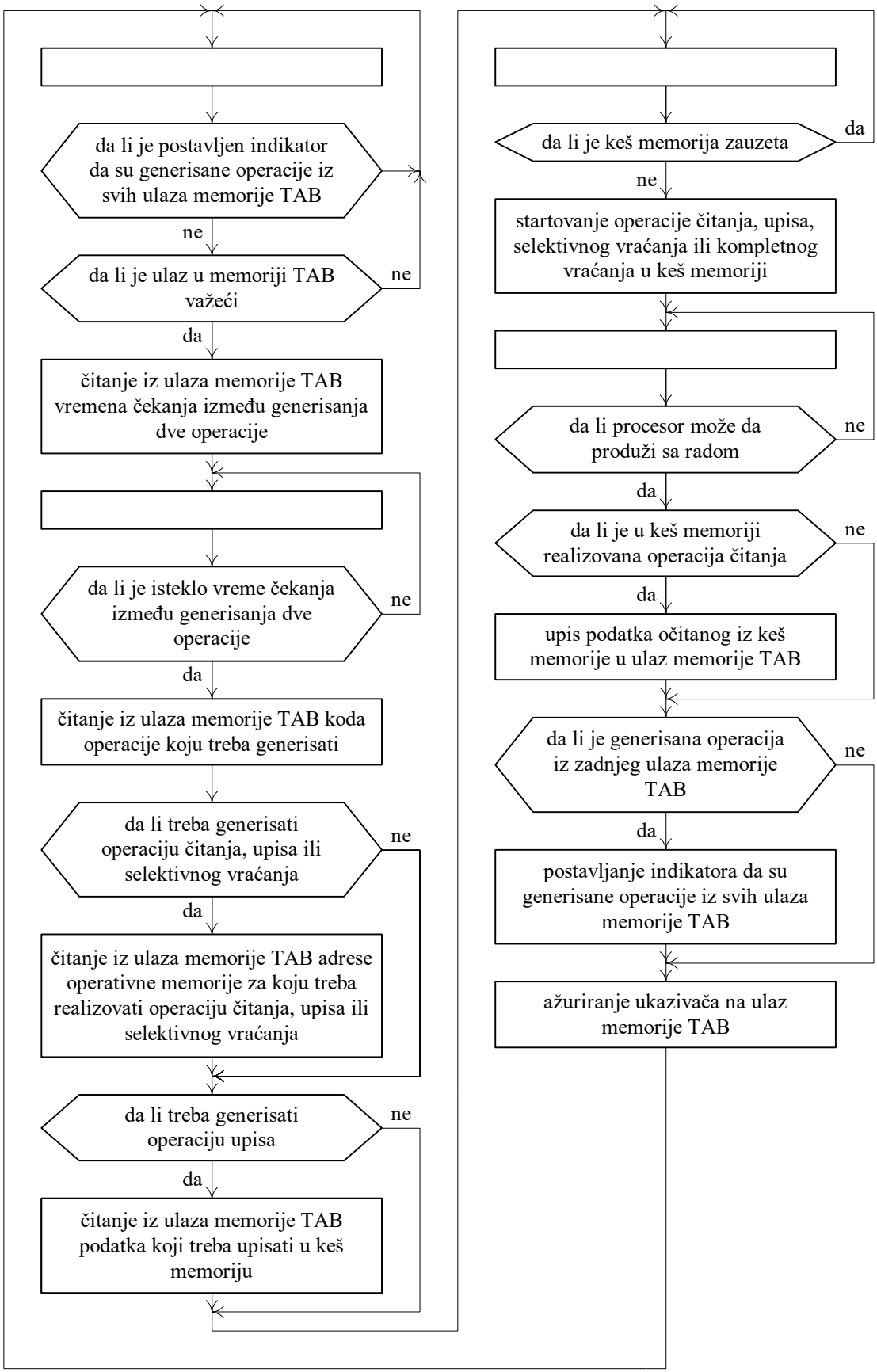
Polja V (Valid), WAIT (WAIT between operations), ADR (ADdRess), DATA (DATA to be written or read) i MODE (MODE of operation) imaju identičnu funkciju kao i polja sa identičnim nazivima u odeljku 1.2.1.1.1.2

1.2.3.1.2. Upravljačka jedinica

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

1.2.3.1.2.1. Dijagram toka operacija

Dijagram toka operacija čitanja, upisa, selektivnog vraćanja i kompletnog vraćanja je dat na slici 65 .



Slika 65 Dijagram toka operacija

U početnom koraku se vrši provera da li je postavljen indikator da su generisane operacije iz svih ulaza memorije TAB. Ukoliko je indikator postavljen ostaje se u početnom koraku i nema generisanja operacija. 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 operacija. Ukoliko je ulaz važeći prelazi se na korake generisanja jedne od četiri operacije. Iz ulaza memorije TAB se, najpre, čita vreme čekanja između generisanja dve operacije, a po isteku ovog vremena i kod operacije koju treba generisati. U slučaju operacija čitanja, upisa i selektivnog vraćanja iz ulaza memorije TAB se čita i adresa operativne memorije za koju treba datu operaciju realizovati, a u slučaju operacije upisa čita se i podatak koji treba upisati u keš memoriju. Potom se vrši provera da li je keš memorija zauzeta. Sve dok je keš memorija zauzeta nema startovanja nove operacije u keš memoriji. Tek kada keš memorija nije zauzeta odgovarajuća operacija se startuje u keš memoriji. Po prijemu indikacije da procesor može da produži sa radom procesor produžava sa preostalim koracima. Najpre se u slučaju da je u keš memoriji startovana operacija čitanja podatak očitan iz keš memorije upisuje u ulaz memorije TAB. Potom se u slučaju da je generisana operacija bila operacija generisana iz zadnjeg ulaza memorije TAB postavlja indikator da su generisane operacije iz svih ulaza memorije TAB. Na kraju se vrši ažuriranje ukazivača na ulaz memorije TAB i prelazi na početni korak.

1.2.3.1.2.2. Sekvenca upravljačkih signala po koracima

Sekvenca upravljačkih signala po koracima formirana na osnovu strukture operacione jedinice (poglavlje 1.2.3.1.1) i dijagrama toka operacija čitanja, upisa, selektivnog vraćanja i kompletnog vraćanja (poglavlje 1.2.3.1.2.1). Notacija koja se koristi je identična sa notacijom iz poglavlja 1.2.1.1.2.2.

Sekvenca upravljačkih signala po koracima je data u daljem tekstu.

! Od koraka $step_0$ se kreće prilikom generisanja svake nove operacije. U korak $step_0$ se dolazi iz koraka $step_6$ po kompletiranju generisanja zadnje operacije. U koraku $step_0$ se proverava da li signali **TV** i $\overline{\textbf{END}}$ bloka *operacije* imaju aktivne vrednosti što se dešava u slučaju kada je odgovarajući ulaz memorije TAB bloka *operacije* važeći i kada još uvek nije generisana operacija za zadnji ulaz memorije TAB, respektivno. U slučaju da signali **TV** i $\overline{\textbf{END}}$ imaju aktivne vrednosti generiše se aktivna vrednost upravljačkog signala **ldWCNT** bloka *operacije*. Signalom **ldWCNT** trajanja jedna perioda signala takta se na signal takta u brojač WCNT upisuje vrednost iz polja WAIT adresiranog ulaza memorije TAB i prelazi na korak $step_1$. U suprotnom slučaju ostaje se u koraku $step_0$. !

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

! U korak $step_1$ se dolazi iz koraka $step_0$. U ovom koraku se vrši provera vrednosti signala **OUT** bloka *operacije*. Neaktivna vrednost signala **OUT** označava da brojač WCNT još uvek nije došao do nule. U tom slučaju se generiše aktivna vrednost signala **decWCNT** bloka *operacije* 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 **ldMODE** bloka *kešinterfejs* trajanja jedna perioda signala takta i njome u registar MODE upisuje vrednost iz polja MODE ulaza memorije TAB bloka *operacije*. Pri aktivnoj vrednosti signala **OUT** prelazi se na korak $step_2$. !

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

! U korak $step_2$ se dolazi iz koraka $step_1$. U njemu se vrši provera da li neki od signala ($\overline{\textbf{TMODE}}_1$ · $\overline{\textbf{TMODE}}_0$), ($\overline{\textbf{TMODE}}_1$ · $\textbf{TMODE}_0$) i ($\textbf{TMODE}_1$ · $\overline{\textbf{TMODE}}_0$) kojima se specificiraju operacije čitanja, upisa i selektivnog vraćanja, respektivno, ima aktivnu vrednost. Ovi signali se formiraju na osnovu signala $\textbf{TMODE}_1$ i $\textbf{TMODE}_0$ bloka *operacije*. Ukoliko jeste, generiše se aktivna vrednost signala **ldPAR** trajanja jedna perioda signala takta kojom se u registar PAR bloka *kešinterfejs* upisuje vrednost iz polja ADR ulaza memorije TAB bloka *operacije*.

Ukoliko je aktivna vrednost signala ($\overline{\text{TMODE}}_1 \cdot \textbf{TMODE}_0$) radi se o operaciji upisa, pa se generiše aktivna vrednost signala **ldPDRWR** trajanja jedna perioda signala takta kojom se u registar PDRWR bloka *kešinterfejs* upisuje vrednost iz polja DATA ulaza memorije TAB. U koraku step_2 se ostaje samo jedna perioda signala takta i bezuslovno se prelazi u korak step_3. !

step_2: *if* ($\overline{\text{TMODE}}_1 \cdot \overline{\text{TMODE}}_0 + \overline{\text{TMODE}}_1 \cdot \textbf{TMODE}_0 + \textbf{TMODE}_1 \cdot \overline{\text{TMODE}}_0$) *then* **ldPAR**,
if ($\overline{\text{TMODE}}_1 \cdot \textbf{TMODE}_0$) *then* **ldPDRWR**,
br step_3

! U korak step_3 se dolazi iz koraka step_2. U ovom koraku se generiše aktivna vrednost signala **PRQ** bloka *kešinterfejs* trajanja jedna perioda signala takta ukoliko je neaktivan signal **BUSY** koji preko bloka *kešinterfejs* keš memorija ***KEŠ*** šalje procesoru ***CPU***. Time se u zavisnosti od sadržaja registra MODE u bloku *kešinterfejs* generiše aktivna vrednost jednog od signala **PRQRD**, **PRQWR**, **PRQSC** i **PRQCC** i time startuje keš memorija ***KEŠ*** da izvrši jednu od operacija čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja, respektivno. U koraku step_3 se ostaje jedna ili više perioda signala takta u zavisnosti od toga da li je pri prelasku na korak step_3 signal **BUSY** neaktivan ili aktivan, respektivno. Pri neaktivnoj vrednosti signala **BUSY** se prelazi u korak step_4. !

step_3: *if* $\overline{\text{BUSY}}$ *then* **PRQ**,
br (*if* $\overline{\text{BUSY}}$ *then* step_4 *else* step_4)

! U korak step_4 se dolazi iz koraka step_3. U koraku step_4 se vrši provera signala **CRP** koji preko bloka *kešinterfejs* keš memorija ***KEŠ*** šalje procesoru ***CPU***. Neaktivna vrednost signala **CRP** je indikacija da procesor ***CPU*** mora da čeka na keš memoriju ***KEŠ***, dok je **a**ktivna vrednost signala **CRP** indikacija da procesor ***CPU*** može da produži sa radom. U sučaju operacije čitanja, aktivnom vrednošću signala **CRP** trajanja jedna perioda signala takta se procesoru **CPU** šalje i indikacija da je očitani bajt podatka raspoloživ na linijama CDRRD. Kako je za operaciju čitanja signal **rd** aktivan, pojavljivanjem aktivne vrednosti signala **CRP** trajanja jedna perioda signala takta i signal **LDPDRRD** postaje aktivan, čime se omogućava upisivanje očitanog podatka u registar PDRRD bloka *kešinterfejs*. U koraku step_4 se ostaje sve dok je neaktivna vrednost signal **CRP**. Kada signal **CRP** postane aktivan prelazi se na korak step_5. !

step_4: *br* (*if* **CRP** *then* step_5 *else* step_4)

! U korak step_5 se dolazi iz koraka step_4. U ovom koraku se jedino u slučaju operacije čitanja, kada su signali $\overline{\text{TMODE}}_1$ i $\overline{\text{TMODE}}_0$ bloka *operacije* aktivni, generiše aktivna vrednost signala **wrTAB** bloka *operacije* trajanja jedna perioda signala takta. Signalom **wrTAB** se očitani podatak upisuje iz registra PDRRD bloka *kešinterfejs* u polje DATA ulaza memorije TAB bloka *operacije*. U slučaju preostale tri operacije u ovom koraku se ne generiše ni jedan od upravljačkih signala. U koraku step_5 se ostaje samo jedna perioda signala takta i bezuslovno se prelazi u korak step_6. !

step_5: *if* ($\overline{\text{TMODE}}_1 \cdot \overline{\text{TMODE}}_0$, **wrTAB**),
br step_6

! U korak step_6 se dolazi iz koraka step_5. U ovom koraku se bezuslovno generišu aktivne vrednost signala **incECNT** i **stEND** bloka *operacije* trajanja jedna perioda signala takta. Aktivnom vrednošću signala **incECNT** se inkrementira sadržaj brojača ECNT i time njegova vrednost podešava na prvi sledeći ulaz memorije TAB bloka *operacije*. Aktivnom vrednošću signala **stEND** treba da se u ovom zadnjem koraku generisanja operacije u slučaju kada je generisana operacija 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_6 se ostaje samo jadna perioda signala takta i bezuslovno se prelazi u korak step_0. !

step_6: **incECNT**, **stEND**,
br step_0

1.2.3.1.2.3. Vremenski oblici signala

Vremenski oblici signala za razmatranu realizaciju su veoma slični vremenskim oblicima signala za realizaciju iz odeljka 1.2.1.1.2.3 datim na slikama 10, 11 i 12. Slike 10 i 11 su date sa ciljem da se ilustruju vremenski oblici signala za operacije čitanja, upisa, selektivnog vraćanja i kompletnog vraćanja i da se pokaže kako se zaustavlja generisanje operacija po generisanju operacije iz zadnjeg ulaza memorije TAB bloka *operacije*. Slika 12 je data sa ciljem da se pokaže kako se zaustavlja generisanje operacija kada se u memoriji TAB naiđe na ulaz koji nije važeći. U slučaju realizacije iz odeljka 1.2.1.1.2.3 ne koristi se tehnika *rani start procesora*, pa keš memorija ***KEŠ*** generisanjem aktivne vrednosti signala **CRP**

signalizira procesoru ***CPU*** da je startovana operacija kompletirana i da procesor ***CPU*** može da produži sa radom i eventualno u keš memoriji ***KEŠ*** startuje novu operaciju čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja. U razmatranoj realizaciji koristi se tehnika *rani start procesora*, pa keš memorija ***KEŠ*** generisanjem aktivne vrednosti signala **CRP** samo dozvoljava procesoru ***CPU*** da produži sa radom iako keš memorija ***KEŠ*** možda nije još uvek kompletirala startovanu operaciju. Zbog toga keš memorija ***KEŠ*** šalje procesoru ***CPU*** i signal **BUSY**. Aktivnom vrednošću signala **BUSY** keš memoriji ***KEŠ*** sprečava procesor ***CPU*** da eventualno startuje novu operaciju čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja pre nego što je tekuća operacija kompletirana, dok neaktivnom vrednošću dozvoljava startovanje nove operacije. Zbog toga su vremenski oblici signala za razmatranu realizaciju veoma slični vremenskim oblicima signala za realizaciju iz odeljka 1.2.1.1.2.3 datim na slikama 10, 11 i 12. Jedina razlika je u tome što je u razmatranoj realizaciji u koraku $step_3$ potrebna i neaktivna vrednost signala **BUSY** da bi se generisala aktivna vrednost signala **PRQ** bloka *kešinterfejs* i prešlo na korak $step_4$. Signale **CRP** i **BUSY** procesor ***CPU*** prima od keš memorije ***KEŠ*** preko bloka *kešinterfejs*.

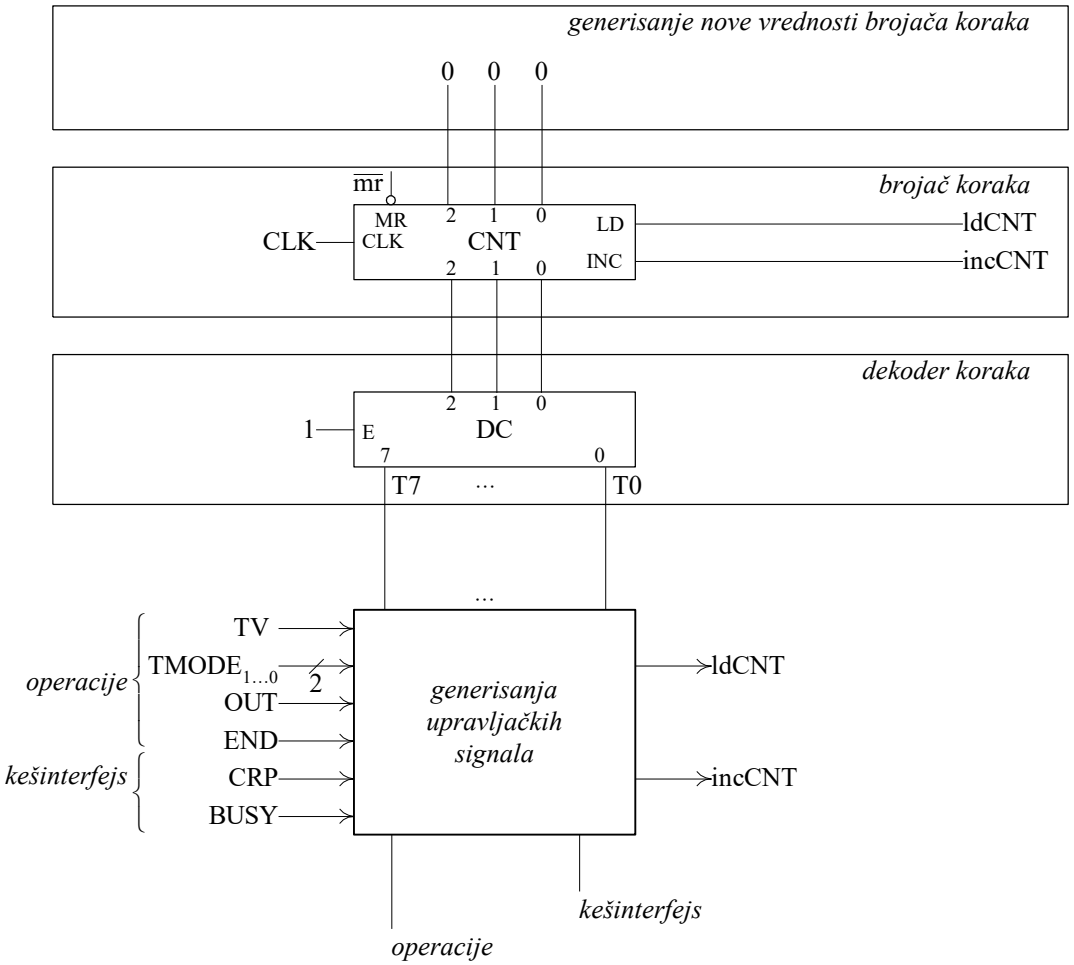
Kao ilustracija uticaja signala **BUSY** na generisanje signala **PRQ** mogu da se koriste vremenski oblici signala dati na slici 42 za realizaciju iz odeljka 1.2.2.1.2.3, jer se i u ovoj realizaciji koristi tehnika *rani start procesora*. Na ovoj slici su ilustrovane dve situacije koje mogu da se jave. Prva situacija se javlja ukoliko je signal **BUSY** već neaktivan kada se pređe na korak $step_3$. Tada se u koraku $step_3$ ostaje samo jedna perioda signal takta, generiše aktivna vrednost signala **PRQ** i prelazi na korak $step_4$. Na signal takta na koji se prelazi iz koraka $step_3$ na korak $step_4$ generiše se aktivna vrednost signal **BUSY**. Ova situacija je ilustrovana vremenskim oblicima signala za operaciju čitanja za koju je uzeto da se generiše za ulaz 0 memorije TAB bloka *operacije*. Druga situacija se javlja ukoliko je signal **BUSY** još uvek aktivan kada se pređe na korak $step_3$. Tada se u koraku $step_3$ ostaje sve dok signal **BUSY** ne postane neaktivan. Na signal takta na koji signal **BUSY** postaje neaktivan, signal **PRQ** postaje aktivan. Na sledeći signal takta signal **PRQ** postaje neaktivan i prelazi se na korak $step_4$. Na signal takta na koji se prelazi iz koraka $step_3$ na korak $step_4$ generiše se aktivna vrednost signala **BUSY**. Ova situacija je ilustrovana vremenskim oblicima signala za operaciju upisa za koju je uzeto da se generiše za ulaz 9 memorije TAB.

1.2.3.1.2.4. Struktura upravljačke jedinice

Struktura upravljačke jedinice ožičene realizacije je prikazana na slici 66. Upravljačka jedinica 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.



Slika 66 Struktura upravljačke jedinice

1.2.3.1.2.4.1. Blok generisanje nove vrednosti brojača koraka

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 1.2.1.1.2.2) se utvrđuje da je 0 vrednost koju treba upisati u brojač CNT. Ta vrednost je ožičena na ulazima 0, 1 i 2 brojača CNT.

1.2.3.1.2.4.2. Blok brojač koraka

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 1.2.3.1.2.2). 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. 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**.

1.2.3.1.2.4.3. Blok dekoder koraka

Blok *dekoder koraka* sadrži dekoder DC. Na ulaze dekodera DC vode se izlazi brojača CNT. Dekodovana stanja brojača CNT pojavljuju se kao signali **T0** do **T7** na izlazima dekodera DC. Svakom koraku iz algoritma generisanja upravljačkih signala (poglavlje 1.2.3.1.2.2) dodeljen je jedan od ovih signala i to koraku $step_0$ signal **T0**, koraku $step_1$ signal **T1**, itd.

1.2.3.1.2.4.4. Blok generisanje upravljačkih signala

Blok *generisanje upravljačkih signala* sadrži kombinacione mreže koje pomoću signala **T0** do **T7** koji dolaze sa bloka *dekoder koraka* upravljačke jedinice, signala logičkih uslova koji dolaze iz blokova operacione jedinice i saglasno algoritmu generisanja upravljačkih signala (poglavlje 1.2.3.1.2.2) generišu upravljačke signale. Postupak projektovanja kombinacionih mreža je identičan sa postupkom koji je objašnjen u poglavlju 1.2.1.1.2.4.4.

Blok *generisanje upravljačkih signala* generiše dve grupe upravljačkih signala i to:
- upravljačke signale operacione jedinice i
- upravljačke signale upravljačke jedinice.

1.2.3.1.2.4.4.1. Upravljački signali operacione jedinice

Upravljački signali operacione jedinice se daju posebno za svaki blok operacione jedinice i to blok *kešinterfejs* i blok *operacije*.

Upravljački signali bloka *kešinterfejs* se generišu na sledeći način:
ldPDRWR = **T2** · ($\overline{\text{TMODE}}_1$ · **TMODE**$_0$),
ldMODE = **T1** · **OUT**,
ldPAR = **T2** · ($\overline{\text{TMODE}}_1$ · $\overline{\text{TMODE}}_0$) i
PRQ = **T3** · **BUSY**.

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:
TMODE$_{1...0}$ — blok *operacije*,
OUT — blok *operacije* i
BUSY — blok *kešinterfejs*.

Upravljački signali bloka *operacije* se generišu na sledeći način:
ldWCNT = **T0** · **TV** · $\overline{\text{END}}$,
decWCNT = **T1** · $\overline{\text{OUT}}$,
incECNT = **T6**,
wrTAB = **T5** · ($\overline{\text{TMODE}}_1$ · $\overline{\text{TMODE}}_0$) i
stEND = **T6**.

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:
TV — blok *operacije*,

END — blok *operacije*,
OUT — blok *operacije* i
TMODE$_{1...0}$ — blok *operacije*.

1.2.3.1.2.4.4.2. Upravljački signali upravljačke jedinice

Upravljački signali upravljačke jedinice su:

- **ldCNT** — signal paralelnog upisa u brojač CNT i
- **incCNT** — signal inkrementiranja sadržaja brojača CNT.

Ovi upravljački signali se generišu na sledeći način:

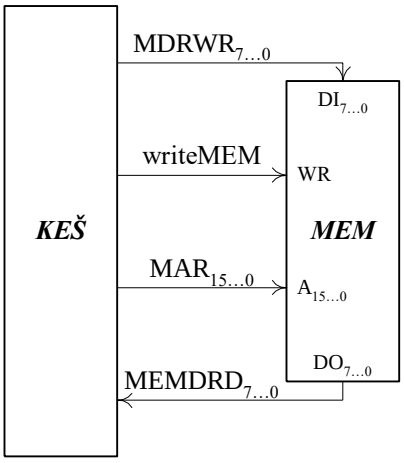
- **ldCNT** = **T6** i
- **incCNT** =**T0** · **TV** · $\overline{\textbf{END}}$ + **T1** · **OUT** + **T2** + **T3** · $\overline{\textbf{BUSY}}$ + **T4** · **CRP** + **T5**.

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

- **TV** — blok *operacije*,
- **END** — blok *operacije*,
- **OUT** — blok *operacije*,
- **BUSY** — blok *kešinterfejs* i
- **CRP** — blok *kešinterfejs*.

1.2.3.2. MEMORIJA MEM

Memoriji ***MEM*** se obraća keš memoriji ***KEŠ*** kod prebacivanja bloka podataka ili iz keš memorije ***KEŠ*** ili iz bafera bloka u memoriju MEM kada treba realizovati operacije upisa u memoriju ***MEM*** i kod prebacivanja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** kada treba realizovati operacije čitanja iz memorije ***MEM*** (slika 67).

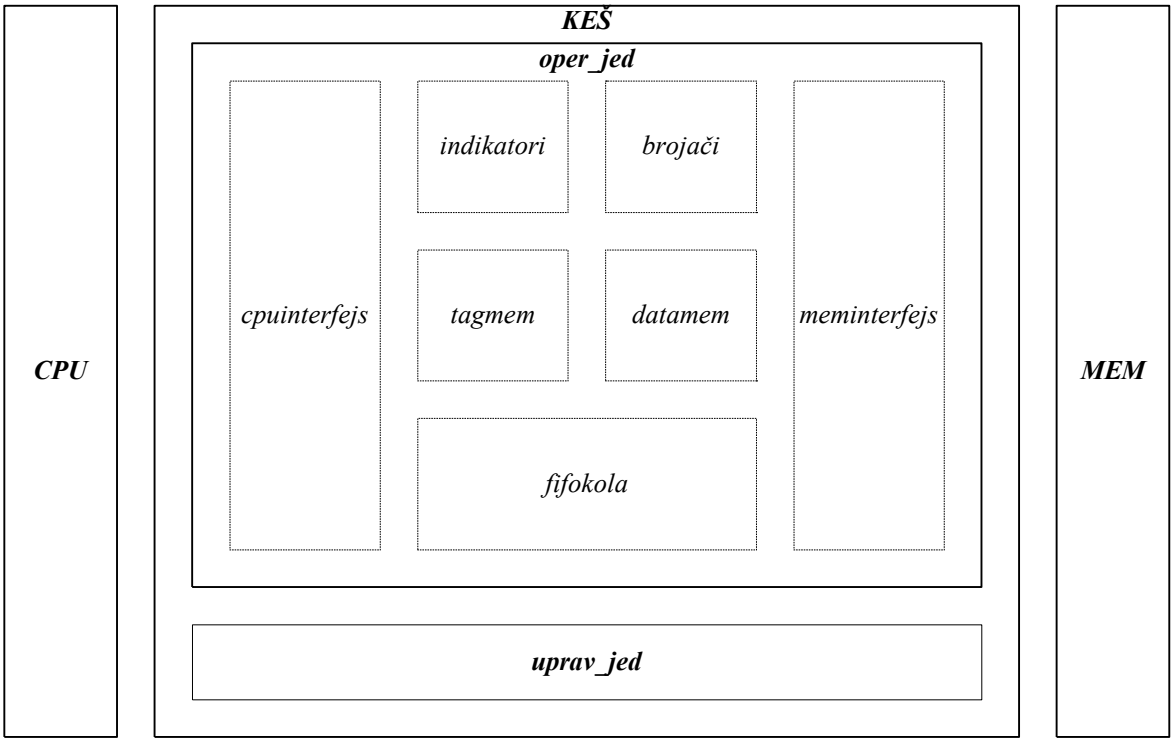


Slika 67 Organizacija memorije ***MEM***

Kod operacije čitanja keš memorija ***KEŠ*** šalje 16-bitnu adresu po linijama MAR$_{15...0}$ i drži neaktivnu vrednost upravljačkog signala **writeMEM**. Očitani 8-bitni podatak se vraća po linijama MEMDRD$_{7...0}$. Kod operacije upisa keš memorija ***KEŠ*** šalje 16-bitnu adresu po linijama MAR$_{15...0}$, 8-bitni podatak po linijama MDRWR$_{7...0}$ i drži aktivnu vrednost signala **writeMEM**. Uzeto je da je vreme pristupa memorije ***MEM*** trajanja deset perioda signala takta.

1.2.3.3. KEŠ MEMORIJA KEŠ

Procesor ***KEŠ*** se sastoji iz operacione jedinice ***oper_ jed*** i upravljačke jedinice ***uprav_ jed*** (slika 68). Operaciona jedinica ***oper_jed*** je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za pamćenje binarnih reči, izvršavanje mikrooperacija i generisanje signala logičkih uslova. Upravljačka jedinica ***uprav_jed*** je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za generisanje upravljačkih signala prema algoritmu generisanja upravljačkih signala operacija keš memorije ***KEŠ*** i signala logičkih uslova. Struktura i opis operacione jedinice i upravljačke jedinice se daju u daljem tekstu.



Slika 68 Organizacija keš memorije ***KEŠ***

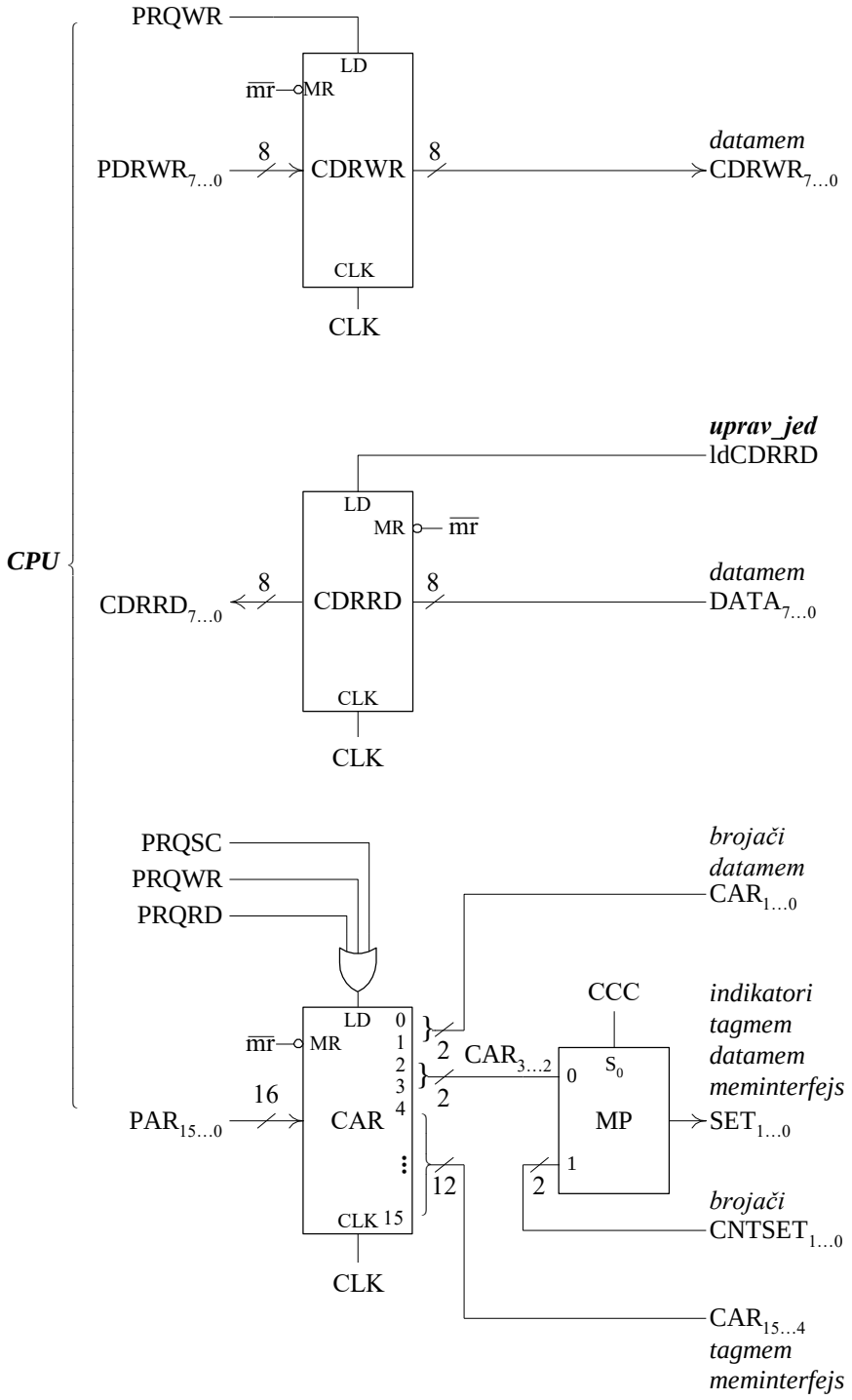
1.2.3.3.1. Operaciona jedinica

Operaciona jedinica ***oper_ jed*** (slika 68) se sastoji od sledećih blokova: blok *cpuinterfejs*, blok *indikatori*, blok *brojači*, blok *tagmem*, blok *datamem*, blok *fifokola* i blok *meminterfejs*. Blok *cpuinterfejs* služi za razmenu adresa, podataka i upravljačkih signala pri radu sa procesorom ***CPU***. Blok *indikatori* služi za vođenje evidencije za 2 ulaza u sva 4 seta keš memorije ***KEŠ*** o tome da li je ulaz zauzet i da li je sadržaj bloka koji se nalazi u ulazu modifikovan nekim prethodnim upisom. Blok *brojači* služi za generisanje adresa bajtova unutar bloka, za brojanje bajtova bloka kod prenosa bloka, za odbrojavanje četiri periode signala takta koliko je vreme pristupa memoriji ***MEM*** i za generisanje broja seta prilikom operacije kompletnog vraćanja. Blok *tagmem* služi za vođenje evidencije za 2 ulaza u sva 4 seta keš memorije o tome kojoj grupi pripada blok podataka koji se nalazi u datom ulazu keš memorije ***KEŠ*** i za privremeno pamćenje broja grupe kojoj pripada blok podataka koji u slučaju operacije čitanja, ukoliko nema saglasnosti u keš memoriji ***KEŠ***, treba vratiti u memoriju ***MEM***. Blok *datamem* služi za smeštanje 8 blokova podataka i za privremeno smeštanje bloka podataka koji u slučaju operacije čitanja, ukoliko nema saglasnosti u keš

memoriji ***KEŠ***, treba vratiti u memoriju ***MEM***. Blok *fifokola* služi za realizaciju FIFO algoritma zamene bloka u setu. Blok *meminterfejs* služi za razmenu adresa, podataka i upravljačkih signala pri radu sa memorijom ***MEM***.

1.2.3.3.1.1. Blok cpuinterfejs

Blok *cpuinterfejs* sadrži registar CAR sa multiplekserom MP, registre CDRWR i CDRRD (slika 69) i flip-flopove CRD, CWR, CSC i CCC (slika 70).



Slika 69 Blok cpuinterfejs (prvi deo)

Registar $CAR_{15...4}$ (Cache Address Register) služi za čuvanje ili 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 ili adrese lokacije memorije ***MEM*** koja pripada bloku koji treba, ukoliko je modifikovan, vratiti iz keš memorije ***KEŠ*** u memoriju ***MEM*** u slučaju operacije selektivnog vraćanja. Na ulaze registra $CAR_{15...4}$ dovodi se 16-bitna adresa iz procesora ***CPU*** po linijama $\mathbf{PAR_{15...0}}$. Ova adresa se upisuje u registar $CAR_{15...4}$ pri pojavi signala takta ukoliko je aktivan jedan od signala **PRQRD, PRQWR** ili **PRQSC** koji dolaze od procesora ***CPU***.

Signali $\mathbf{CAR_{15...4}}$ sa izlaza registra $CAR_{15...4}$ se vode:

- u blok *tagmem* gde se koriste za utvrđivanje saglasnosti kod operacija čitanja, upisa i selektivnog vraćanja, i za upisivanje u memorijski modul TAG0 ili TAG1 kod dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** i
- u blok *meminterfejs* gde se koriste u formiranju adrese memorije ***MEM*** kod dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** i kod vraćanja bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***.

Signali $\mathbf{CAR_{3...2}}$ sa izlaza registra $CAR_{15...4}$ se vode na ulaz multipleksera MP.

Multiplekser MP služi za selekciju signala $\mathbf{CAR_{3...2}}$ i $\mathbf{CNTSET_{1...0}}$ i formiranje signala $\mathbf{SET_{1...0}}$. U slučaju operacija čitanja, upisa ili selektivnog vraćanja signal **CCC** je neaktivan, pa se kroz multiplekser propuštaju signali $\mathbf{CAR_{3...2}}$. U slučaju operacije kompletnog vraćanja signal **CCC** je aktivan, pa se kroz multiplekser propuštaju signali $\mathbf{CNTSET_{1...0}}$. Selektovani signali se kao signali $\mathbf{SET_{1...0}}$ vode u :

- blok *indikatori* radi adresiranja ulaza indikatora V0 i V1 i D0 i D1,
- blok *tagmem* radi adresiranja ulaza memorijskih modula TAG0 i TAG1,
- blok *datamem* gde se koriste u formiranju adrese memorijskih modula DATA0 i DATA1 sa koje treba ili nešto očitati ili na kojoj treba nešto upisati i
- blok *meminterfejs* gde se koriste u formiranju adrese memorije ***MEM*** kod dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** i kod vraćanja bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***.

Signali $\mathbf{CAR_{1...0}}$ sa izlaza registra $CAR_{15...4}$ se vode u :

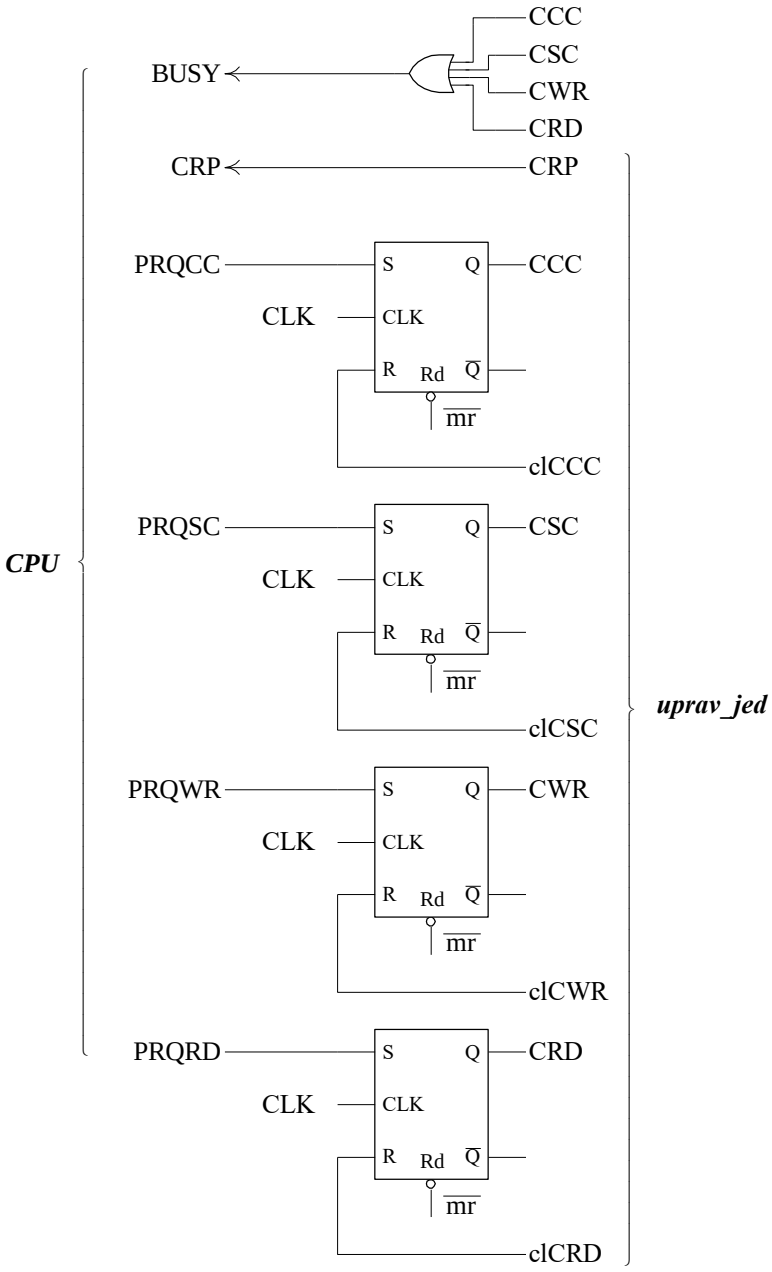
- blok *brojači* gde se pamte kao adresa bajta u bloku kome se prvo pristupa u slučaju operacija čitanja i upisa i
- blok *datamem* gde se koriste za adresiranje reči u bloku pri otkrivenoj saglasnosti u slučaju operacija čitanja i upisa.

Registar $CDRWR_{7...0}$ (Cache Data Register for WRite) služi za čuvanje podatka koji treba upisati u slučaju operacije upisa. Na ulaze registra $CDRWR_{7...0}$ dovodi se po linijama $\mathbf{PDRWR_{7...0}}$ 8-bitni podatak iz procesora ***CPU***. Ovaj podatak se upisuje u registar $CDRWR_{7...0}$ pri pojavi signala takta ukoliko je aktivan signal **PRQWR** koji dolazi od procesora ***CPU***. Sadržaj registra $CDRWR_{7...0}$ se vodi u blok *datamem* u kome se upisuje u memorijski modul DATA0 ili DATA1 u slučaju operacije upisa.

Registar $CDRRD_{7...0}$ (Cache Data Register for ReaD) služi za čuvanje očitanog podatka u slučaju operacije čitanja. Na ulaze registra $CDRRD_{7...0}$ se po linijama $\mathbf{DATA_{7...0}}$ dovodi 8-bitni podatak. Ovaj podatak se upisuje u registar $CDRRD_{7...0}$ pri pojavi signala takta ukoliko je aktivan signal **ldCDRRD**.

Aktivnom vrednošću signala **PRQRD**, **PRQWR**, **PRQSC** ili **PRQCC** trajanja jedna perioda signala takta procesor ***CPU*** signalizira keš memoriji ***KEŠ*** da treba da startuje izvršavanje operacije čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja, respektivno. Aktivnom vrednošću signala **CRP** trajanja jedna perioda signala takta keš

memorija ***KEŠ*** signalizira procesoru ***CPU*** da može da nastavi sa radom bez obzira na to da li je odgovarajuća operacija kompletirana na strani keš memorije ***KEŠ***.



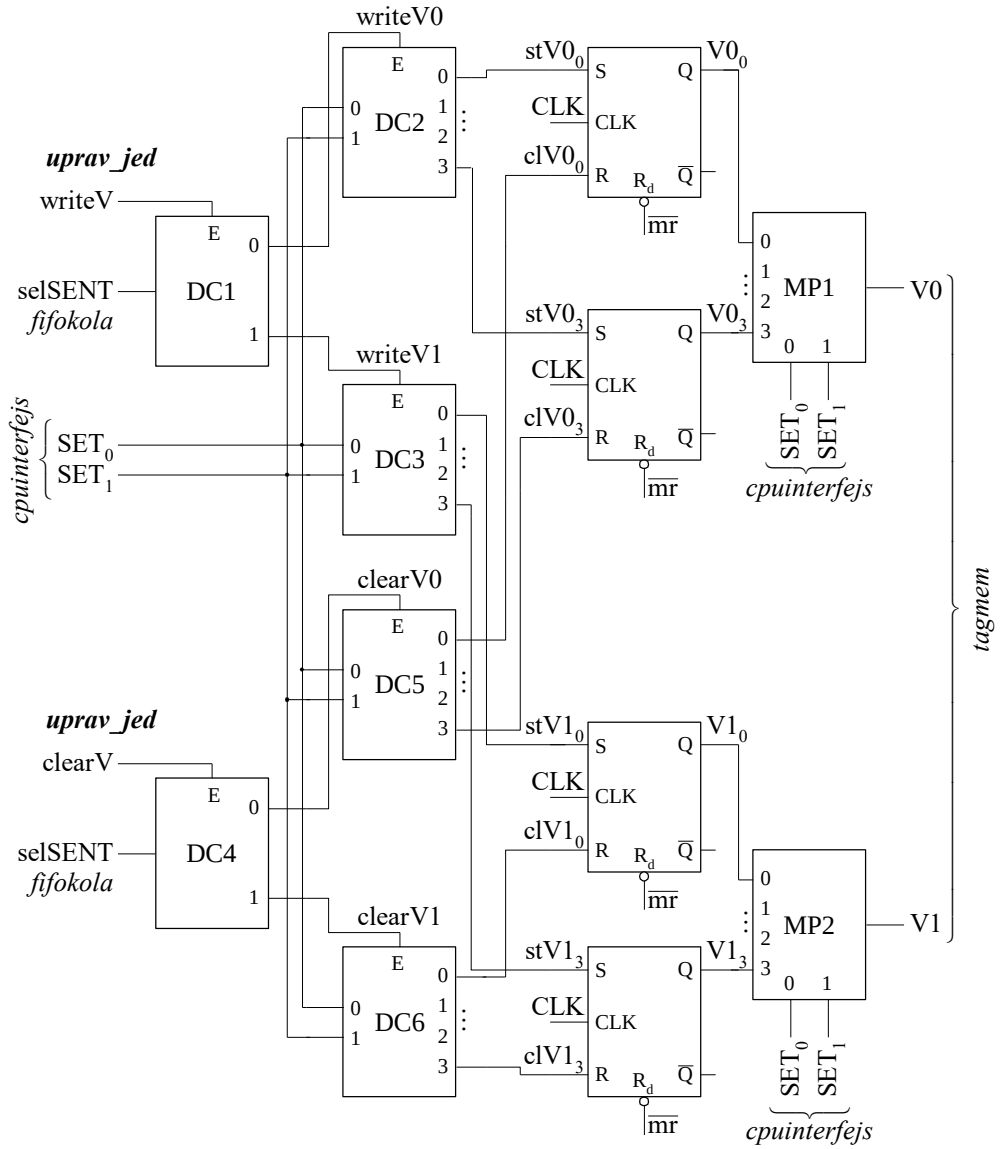
Slika 70 Blok cpuinterfejs (drugi deo)

Flip-flopovi CRD (Cache ReaD), CWR (Cache WRite), CSC (Cache Selectively Clear) i CCC (Cache Completely Clear) služe da se u keš memoriji ***KEŠ*** upamti da su u toku operacije čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja, respektivno. Flip-flop CRD se postavlja na aktivnu vrednost na signal takta pri pojavi aktivne vrednosti signala **PRQRD** koji dolazi iz procesora ***CPU*** i traje jednu periodu signala takta. Aktivna vrednost ostaje sve dok je operacija čitanja u toku. Flip-flop CRD se postavlja na neaktivnu vrednost na kraju operacije čitanja na signal takta pri aktivnoj vrednosti signala **clCRD** trajanja jedna perioda signala takta. Za flip-flopove CWR, CSC i CCC važi sve što je rečeno za flip-flop CRD. Tako se u slučaju operacije upisa flip-flop CWR postavlja na aktivnu vrednost pri pojavi aktivne vrednosti signala **PRQWR** i postavlja na neaktivnu vrednost pri pojavi aktivne vrednosti

signala **clCWR**. U slučaju operacije selektivnog vraćanja flip-flop CSC se postavlja na aktivnu vrednost pri pojavi aktivne vrednosti signala **PRQSC** i postavlja na neaktivnu vrednost pri pojavi aktivne vrednosti signala **clCSC**. U slučaju operacije kompletnog vraćanja flip-flop CCC se postavlja na aktivnu vrednost pri pojavi aktivne vrednosti signala **PRQCC** i postavlja na neaktivnu vrednost pri pojavi aktivne vrednosti signala **clCCC**. Dok je u toku bilo koja od četiri operacije signal **BUSY** je aktivan, a po kompletiranju signal **BUSY** postaje neaktivan.

1.2.3.3.1.2. Blok indikatori

Blok *indikatori* sadrži flip-flopove $V0_0$ do $V0_3$ i $V1_0$ do $V1_3$, dekodere DC1 do DC6, multipleksere MP1 i MP2 (slika 71), flip-flopove $D0_0$ do $D0_3$ i $D1_0$ do $D1_3$, dekodere DC7 do DC12 i multipleksere MP3, MP4 i MP5 (slika 72).



Slika 71 Blok indikatori (prvi deo)

Flip-flopovi $V0_0$ do $V0_3$ i $V1_0$ do $V1_3$ služe za čuvanje indikatora V (Valid). Flip-flopovima $V0_0$ do $V0_3$ se za svaki nulti i flip-flopovima $V1_0$ do $V1_3$ za svaki prvi ulaz sva

četiri seta keš memorije ***KEŠ*** vodi evidencija da li je važeći ili ne. U slučaju operacija čitanja i upisa se prilikom dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** generiše aktivnu vrednost signala **writeV**. Tada se na signal takta upisuje aktivna vrednost u jedan od flip-flopova $V0_0$ do $V0_3$ i $V1_0$ do $V1_3$ i to onaj koji je adresiran signalima **selSENT** i $\mathbf{SET_{1...0}}$. Signalom **selSENT** se selektuju ulazi nula ili jedan, a signalima $\mathbf{SET_{1...0}}$ jedan od četiri seta. U slučaju operacija selektivnog vraćanja i kompletnog vraćanja se prilikom vraćanja bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM*** generiše aktivna vrednost signala **clearV**. Tada se na signal takta upisuje neaktivna vrednost u jedan od flip-flopova $V0_0$ do $V0_3$ i $V1_0$ do $V1_3$ i to onaj koji je adresiran signalima **selSENT** i $\mathbf{SET_{1...0}}$. Signalom **selSENT** se selektuju ulazi nula ili jedan, a signalima $\mathbf{SET_{1...0}}$ jedan od četiri seta.

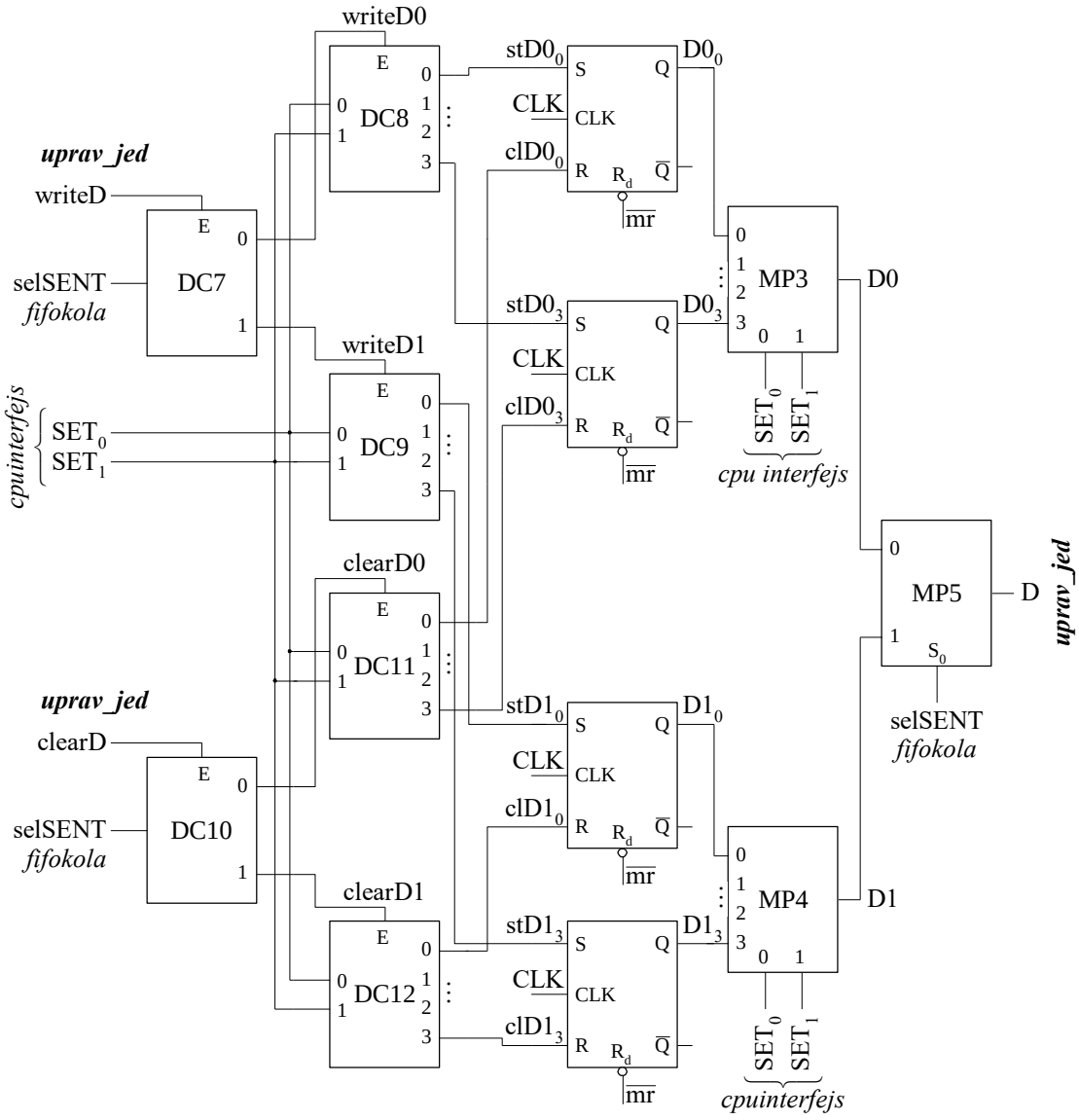
Dekoderi DC1, DC2 i DC3 služe za formiranje signala **writeV0** i **writeV1**, $\mathbf{stV0_0}$ do $\mathbf{stV0_3}$ i $\mathbf{stV1_0}$ do $\mathbf{stV1_3}$, respektivno. Pri aktivnoj vrednosti signala **writeV**, a u zavisnosti od vrednosti signala **selSENT**, generiše se aktivna vrednost signala **writeV0** ili **writeV1**. Time se vrši selekcija između flip-flopova ulaza nula $V0_0$ do $V0_3$ ili flip-flopova ulaza jedan $V1_0$ do $V1_3$ za upisivanje aktivne vrednosti. Pri aktivnoj vrednosti signala **writeV0**, a u zavisnosti od vrednosti signala $\mathbf{SET_{1...0}}$, generiše se aktivna vrednost jednog od signala $\mathbf{stV0_0}$ do $\mathbf{stV0_3}$ i time određuje jedan od flip-flopova ulaza nula $V0_0$ do $V0_3$ u koji se upisuje aktivna vrednost. Pri aktivnoj vrednosti signala **writeV1**, a u zavisnosti od vrednosti signala $\mathbf{SET_{1...0}}$, generiše se aktivna vrednost jednog od signala $\mathbf{stV1_0}$ do $\mathbf{stV1_3}$ i time određuje jedan od flip-flopova ulaza jedan $V1_0$ do $V1_3$ u koji se upisuje aktivna vrednost.

Dekoderi DC4, DC5 i DC6 služe za formiranje signala **clearV0** i **clearV1**, $\mathbf{clV0_0}$ do $\mathbf{clV0_3}$ i $\mathbf{clV1_0}$ do $\mathbf{clV1_3}$, respektivno. Pri aktivnoj vrednosti signala **clearV**, a u zavisnosti od vrednosti signala **selSENT**, generiše se aktivna vrednost signala **clearV0** ili **clearV1**. Time se vrši selekcija između flip-flopova ulaza nula $V0_0$ do $V0_3$ ili flip-flopova ulaza jedan $V1_0$ do $V1_3$ za upisivanje neaktivne vrednosti. Pri aktivnoj vrednosti signala **clearV0**, a u zavisnosti od vrednosti signala $\mathbf{SET_{1...0}}$, generiše se aktivna vrednost jednog od signala $\mathbf{clV0_0}$ do $\mathbf{clV0_3}$ i time određuje jedan od flip-flopova ulaza nula $V0_0$ do $V0_3$ u koji se upisuje neaktivna vrednost. Pri aktivnoj vrednosti signala **clearV1**, a u zavisnosti od vrednosti signala $\mathbf{SET_{1...0}}$, generiše se aktivna vrednost jednog od signala $\mathbf{clV1_0}$ do $\mathbf{clV1_3}$ i time određuje jedan od flip-flopova ulaza jedan $V1_0$ do $V1_3$ u koji se upisuje neaktivna vrednost.

Multiplekseri MP1 i MP2 služe za formiranje signala **V0** i **V1**. Signal **V0** u slučaju operacija čitanja, upisa, selektivnog vraćanja i kompletnog vraćanja predstavlja indikaciju da li je ulaz nula određenog seta keš memorije ***KEŠ*** važeći. Signal **V1** u slučaju operacija čitanja, upisa, selektivnog vraćanja i kompletnog vraćanja predstavlja indikaciju da li je ulaz jedan određenog seta keš memorije ***KEŠ*** važeći. Na informacione ulaze multipleksera MP1 vode se signali $\mathbf{V0_0}$ do $\mathbf{V0_3}$, na informacione ulaze multipleksera MP2 signali $\mathbf{V1_0}$ do $\mathbf{V1_3}$, a na upravljačke ulaze oba multipleksera signali $\mathbf{SET_{1...0}}$. Signali $\mathbf{SET_{1...0}}$ su formirani od signala $\mathbf{CAR_{3...2}}$ generisane adrese u slučaju operacija čitanja, upisa i selektivnog vraćanja i signala $\mathbf{CNTSET_{1...0}}$ brojača $CNTSET_{1...0}$ u slučaju operacije kompletnog vraćanja.

Flip-flopovi $D0_0$ do $D0_3$ i $D1_0$ do $D1_3$ služe za čuvanje indikatora D (Dirty). Flip-flopovima $D0_0$ do $D0_3$ se za svaki nulti ulaz i flip-flopovima $D1_0$ do $D1_3$ za svaki prvi ulaz sva četiri seta keš memorije ***KEŠ*** vodi evidencija da li je modifikovan ili ne. Ukoliko se u slučaju operacije upisa otkrije saglasnost i upis izvrši u keš memoriju ***KEŠ***, generiše se aktivna vrednost signala **writeD**. Tada se na signal takta upisuje aktivna vrednost u jedan od flip-flopova $D0_0$ do $D0_3$ i $D1_0$ do $D1_3$ i to onaj koji adresiran signalima **selSENT** i $\mathbf{SET_{1...0}}$. Signalom **selSENT** se selektuju ulazi nula ili jedan, a signalima $\mathbf{SET_{1...0}}$ jedan od četiri seta. Ukoliko u slučaju operacije čitanja ili upisa nema saglasnosti, za zamenu se bira jedan od dva ulaza seta određenog generisanom adresom. Ako je dati ulaz modifikovan, blok iz datog ulaza se najpre vraća iz keš memorije ***KEŠ*** u memoriju ***MEM***, zatim se dovlači novi blok iz

memorije ***MEM*** u dati ulaz keš memorije ***KEŠ***. Ako dati ulaz nije modifikovan, odmah se dovlači novi blok iz memorije ***MEM*** u dati ulaz keš memorije ***KEŠ***. Po dovlačenju novog bloka generiše se aktivna vrednost signala **clearD**. Ukoliko u slučaju operacije selektivnog vraćanja postoji saglasnost u jednom od dva ulaza seta određenog generisanom adresom, blok iz datog ulaza se vraća ili ne vraća iz keš memorije ***KEŠ*** u memoriju ***MEM*** u zavisnosti od toga da li je modifikovan ili nije i generiše se aktivna vrednost signala **clearD**. Ukoliko je u slučaju operacije kompletnog vraćanja ulaz u kome se vrši provera modifikovan, blok iz datog ulaza se vraća iz keš memorije ***KEŠ*** u memoriju ***MEM*** i generiše se aktivna vrednost signala **clearD**. U svim situacijama u kojima se generiše aktivna vrednost signala **clearD**, na signal takta se upisuje neaktivna vrednost u jedan od flip-flopova $D0_0$ do $D0_3$ i $D1_0$ do $D1_3$ i to onaj koji adresiran signalima **selSENT** i $\mathbf{SET_{1...0}}$. Signalom **selSENT** se selektuju ulazi nula ili jedan, a signalima $\mathbf{SET_{1...0}}$ jedan od četiri seta.



Slika 72 Blok indikatori (drugi deo)

Dekoderi DC7, DC8 i DC9 služe za formiranje signala **writeD0** i **writeD1**, $\mathbf{stD0_0}$ do $\mathbf{stD0_3}$ i $\mathbf{stD1_0}$ do $\mathbf{stD1_3}$, respektivno. Pri aktivnoj vrednosti signala **writeD**, a u zavisnosti od vrednosti signala **selSENT**, generiše se aktivna vrednost signala **writeD0** ili **writeD1**. Time

se vrši selekcija između flip-flopova ulaza nula $D0_0$ do $D0_3$ ili flip-flopova ulaza jedan $D1_0$ do $D1_3$ za upisivanje aktivne vrednosti. Pri aktivnoj vrednosti signala **writeD0**, a u zavisnosti od vrednosti signala $\mathbf{SET_{1...0}}$, generiše se aktivna vrednost jednog od signala $\mathbf{stD0_0}$ do $\mathbf{stD0_3}$ i time određuje jedan od flip-flopova ulaza nula $D0_0$ do $D0_3$ u koji se upisuje aktivna vrednost. Pri aktivnoj vrednosti signala **writeD1**, a u zavisnosti od vrednosti signala $\mathbf{SET_{1...0}}$, generiše se aktivna vrednost jednog od signala $\mathbf{stD1_0}$ do $\mathbf{stD1_3}$ i time određuje jedan od flip-flopova ulaza jedan $D1_0$ do $D1_3$ u koji se upisuje aktivna vrednost.

Dekoderi DC10, DC11 i DC12 služe za formiranje signala **clearD0** i **clearD1**, $\mathbf{clD0_0}$ do $\mathbf{clD0_3}$ i $\mathbf{clD1_0}$ do $\mathbf{clD1_3}$, respektivno. Pri aktivnoj vrednosti signala **clearD**, a u zavisnosti od vrednosti signala **selSENT**, generiše se aktivna vrednost signala **clearD0** ili **clearD1**. Time se vrši selekcija između flip-flopova ulaza nula $D0_0$ do $D0_3$ ili flip-flopova ulaza jedan $D1_0$ do $D1_3$ za upisivanje neaktivne vrednosti. Pri aktivnoj vrednosti signala **clearD0**, a u zavisnosti od vrednosti signala $\mathbf{SET_{1...0}}$, generiše se aktivna vrednost jednog od signala $\mathbf{clD0_0}$ do $\mathbf{clD0_3}$ i time određuje jedan od flip-flopova ulaza nula $D0_0$ do $D0_3$ u koji se upisuje neaktivna vrednost. Pri aktivnoj vrednosti signala **clearD1**, a u zavisnosti od vrednosti signala $\mathbf{SET_{1...0}}$, generiše se aktivna vrednost jednog od signala $\mathbf{clD1_0}$ do $\mathbf{clD1_3}$ i time određuje jedan od flip-flopova ulaza jedan $D1_0$ do $D1_3$ u koji se upisuje neaktivna vrednost.

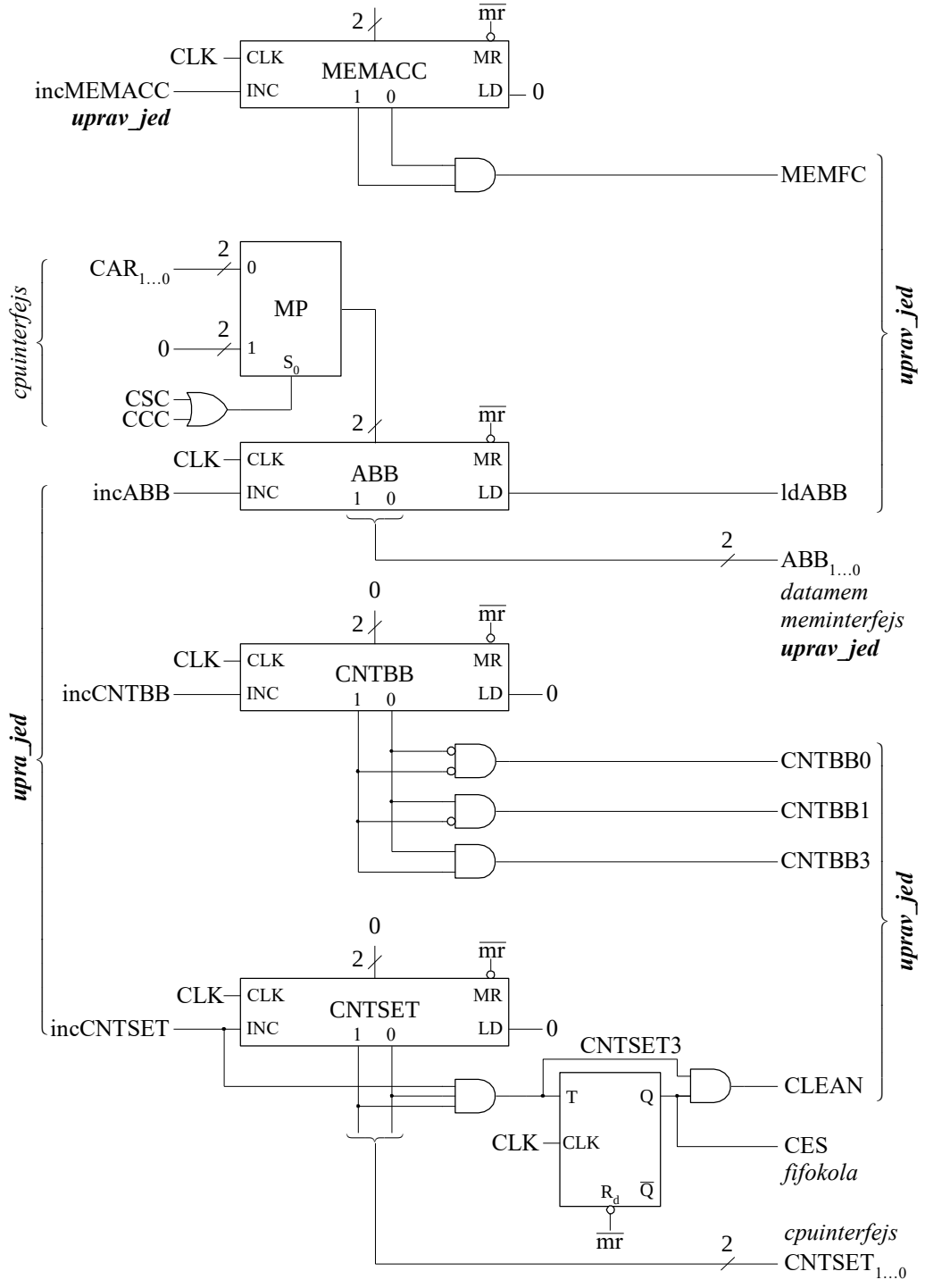
Multiplekseri MP3 i MP4 služe za formiranje signala **D0** i **D1**. Signal **D0** u slučaju operacija čitanja, upisa, selektivnog vraćanja i kompletnog vraćanja predstavlja indikaciju da li je ulaz nula određenog seta keš memorije ***KEŠ*** modifikovan. Signal **D1** u slučaju operacija čitanja, upisa, selektivnog vraćanja i kompletnog vraćanja predstavlja indikaciju da li je ulaz jedan određenog seta keš memorije ***KEŠ*** modifikovan. Na informacione ulaze multipleksera MP3 vode se signali $\mathbf{D0_0}$ do $\mathbf{D0_3}$, na informacione ulaze multipleksera MP4 signali $\mathbf{D1_0}$ do $\mathbf{D1_3}$, a na upravljačke ulaze oba multipleksera signali $\mathbf{SET_{1...0}}$. Signali $\mathbf{SET_{1...0}}$ su formirani od signala $\mathbf{CAR_{3...2}}$ generisane adrese u slučaju operacija čitanja, upisa i selektivnog vraćanja i signala $\mathbf{CNTSET_{1...0}}$ brojača $CNTSET_{1...0}$ u slučaju operacije kompletnog vraćanja.

Multiplekser MP5 služi za formiranje signala **D1**. Na informacione ulaze multipleksera MP5 vode se signali **D0** i **D1**, a na upravljački ulaz signal **selSENT**.

1.2.3.3.1.3. Blok brojači

Blok *brojači* (slika 73) sadrži brojač MEMACC, brojač ABB sa multiplekserom MP, brojač CNTBBB i brojač CNTSET sa flip-flopom CES.

Brojač $MEMACC_{1..0}$ (MEMory ACCess time) je brojač vremena pristupa operativnoj memoriji koji se koristi kod čitanja iz memorije ***MEM*** ili upisa u memoriju ***MEM*** da odbroji četiri signala takta koliko je usvojeno da je vreme pristupa memoriji ***MEM***. Početna vrednost brojača je nula. Pri svakom obraćanju memoriji ***MEM*** generiše se aktivna vrednosti signala **incMEMACC** pa se na signal takta vrši inkrementiranje brojača. Kada brojač pređe u stanje tri signal **MEMFC** (MEMory Function Completed) postaje aktivan. Na sledeći signal takta vrši se četvrto inkrementiranje, pa se brojač $MEMACC_{1..0}$ vraća u početno stanje nula. Zbog toga je signal **MEMFC** aktivan u trajanju jedna perioda signala takta. Signal **MEMFC** se koristi i kao signal logičkog uslova i njegova vrednost jedan je indikacija da je pristup memoriji završen.



Slika 73 Blok brojači

Brojač $ABB_{1..0}$ (Address of Block Bytes) je brojač adresa bajtova bloka koji se koristi za generisanje adresa bajtova u bloku kod dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** i kod vraćanja bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***. Brojač $ABB_{1..0}$ se koristi u bloku *datamem* kod formiranja adrese memorijskih modula DATA0, DATA1 i DATABUF i u bloku *meminterfejs* kod formiranja adresa memorije ***MEM***. U brojač $ABB_{1..0}$ se na signal takta i pri aktivnoj vrednosti signala **ldABB** upisuje vrednost sa izlaza multipleksera MP. U slučaju operacije čitanja ili upisa to je vrednost sa adresnih linija $CAR_{1...0}$, jer su tada signali **CSC** i **CCC** neaktivni. U slučaju operacija

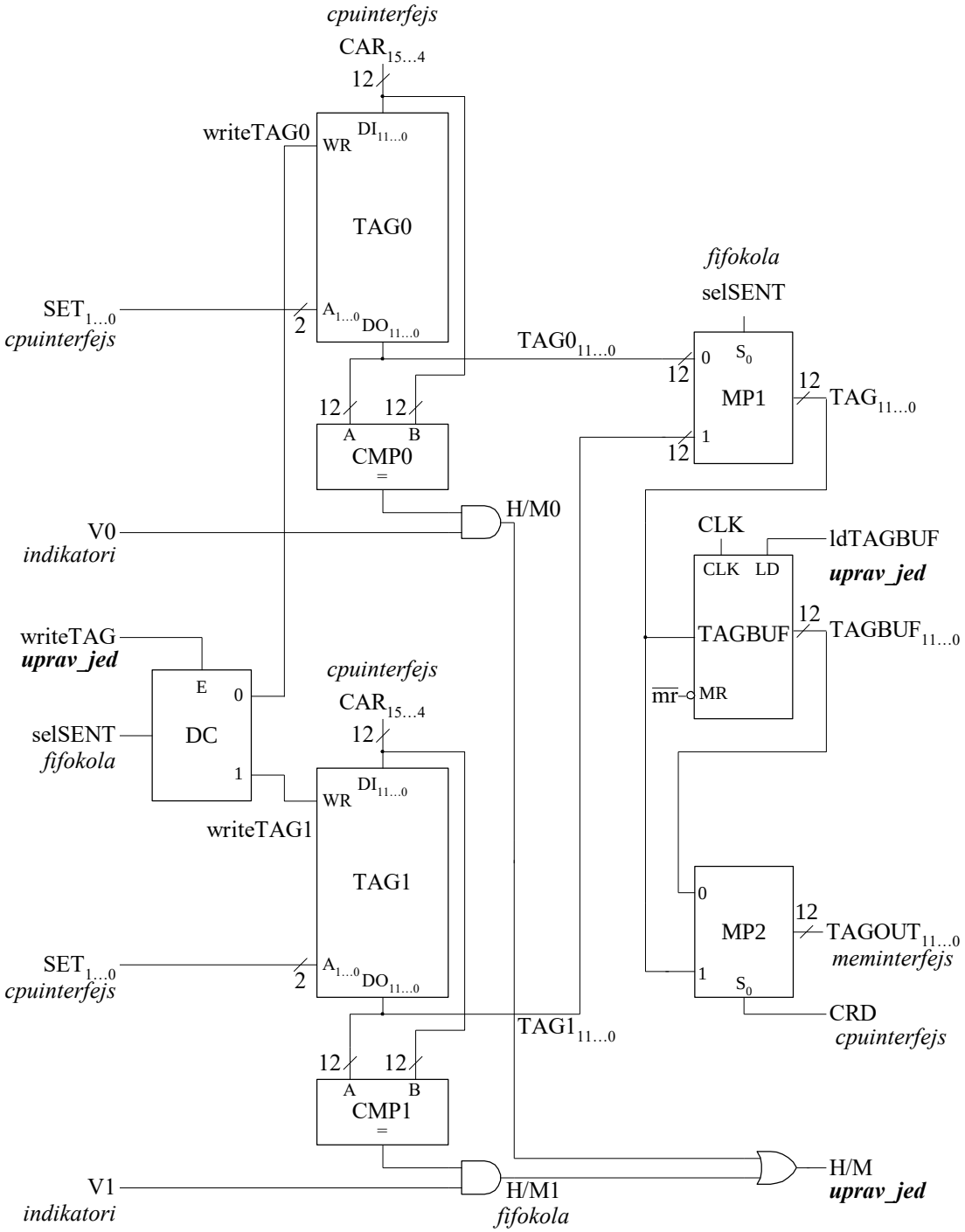
selektivnog vraćanja i kompletnog vraćanja to je nula, jer je tada jedan od signala **CSC** ili **CCC** aktivan. Sadržaj brojača $ABB_{1..0}$ se na signal takta i pri aktivnoj vrednosti signala **incABB** inkrementira. Ovim se obezbeđuje da se pri operacijama čitanja i upisa prvo pristupa adresiranom a zatim ostalim bajtovima bajtu u bloku, a u slučaju operacija selektivnog vraćanja i kompletnog vraćanja bajtovima u bloku se pristupa redom od nultog do trećeg bajta.

Brojač $CNTBB_{1..0}$ (CouNTer of Bytes of Blocks) je brojač bajtova bloka koji se koristi za brojanje prenetih bajtova bloka kod dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** i kod vraćanja bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***. Početna vrednost brojača je nula. Sadržaj brojača $CNTBB_{1..0}$ se na signal takta i pri aktivnoj vrednosti signala **incCNTBB** inkrementira. Posle četvrtog inkrementiranja brojač se vraća u početno stanje nula. Priliom prolaska brojača kroz stanja nula, jedan i tri generišu se aktivne vrednosti signala **CNTBB0**, **CNTBB1** i **CNTBB3**, respektivno, koji se koriste kao signali logičkih uslova.

Brojač $CNTSET_{1..0}$ (CouNTer of cache SET) je brojač setova u keš memoriji koji se koristi zajedno sa flip-flopom CES (Cache Entry Selection) selekcije ulaza nula i ulaza jedan u keš memoriji pri izvršavanju operacije kompletno vraćanje. Brojač $CNTSET_{1..0}$ ukazuje na set keš memorije a flip-flop CES na ulaz seta za koji se trenutno realizuje operacija kompletno vraćanje. Početne vrednosti flip-flopa CES i brojača $CNTSET_{1..0}$ su nula. S toga se operacija kompletno vraćanje najpre realizuje za blok iz ulaza nula seta nula. Pri aktivnoj vrednosti signala **incCNTSET** se na signal takta vrši inkrementiranje brojača $CNTSET_{1..0}$. S toga se operacija kompletno vraćanje sada realizuje za blok iz ulaza nula seta jedan. Na sličan način se vrše još dva inkrementiranja brojača $CNTSET_{1..0}$ i operacija kompletno vraćanje za blokove iz ulaza nula setova dva i tri. Brojač $CNTSET_{1..0}$ je sada u stanju tri, pa je pri sledećoj aktivnoj vrednosti signala **incCNTSET** na ulazu flip-flopa CES aktivna vrednost. Na prvi signal takta se brojač $CNTSET_{1..0}$ prebacuje is stanja tri u stanje nula, a flip-flop CES postavlja na vrednost jedan. S toga se operacija kompletno vraćanje sada realizuje za blok iz ulaza jedan seta nula. Pri aktivnoj vrednosti signala **incCNTSET** se na signal takta vrši inkrementiranje brojača $CNTSET_{1..0}$. S toga se operacija kompletno vraćanje sada realizuje za blok iz ulaza jedan seta jedan. Na sličan način se vrše još dva inkrementiranja brojača $CNTSET_{1..0}$ i operacija kompletno vraćanje za blokove iz ulaza jedan setova dva i tri. Brojač $CNTSET_{1..0}$ je sada u stanju tri, pa je pri sledećoj aktivnoj vrednosti signala **incCNTSET** na ulazu flip-flopa CES aktivna vrednost. Pored toga sada je i signal **CLEAN** postao aktivan.Na prvi signal takta se brojač $CNTSET_{1..0}$ prebacuje is stanja tri u stanje nula, flip-flop CES postavlja na vrednost nula i signal **CLEAN** postaje neaktivan. Aktivna vrednost signala **CLEAN** trajanja jedna perioda signala takta je indikacija da je operacija kompletno vraćanje realizovana za sve blokove keš memorije.

1.2.3.3.1.4. Blok tagmem

Blok *tagmem* (slika 74) sadrži memorijske module TAG0 i TAG1 sa komparatorima CMP0 i CMP1, dekoder DC, multiplekser MP1, registar TAGBUF i multiplekser MP2.



Slika 74 Blok tagmem

Memorijski moduli TAG0 i TAG1 služe za čuvanje brojeva grupa za nulte i prve ulaze četiri seta keš memorije, respektivno, a kapacitet im je po 4 reči širine 12 bita. Adresiranje se vrši signalima $\mathbf{SET_{1..0}}$ koji se vode na ulazne linije $A_{1...0}$ memorijskih modula TAG0 i TAG1 i predstavljaju broj seta u keš memoriji ***KEŠ***. U slučaju operacija čitanja, upisa i selektivnog vraćanja signali $\mathbf{SET_{1..0}}$ se formiraju od signala $\mathbf{CAR_{3...2}}$ generisane adrese. U slučaju operacije kompletnog vraćanja signali $\mathbf{SET_{1..0}}$ se formiraju od signala $\mathbf{CNTSET_{1...0}}$ brojača $CNTSET_{1..0}$. Očitane vrednosti se pojavljuje kao signali $\mathbf{TAG0_{11...0}}$ na izlaznim linijama podataka $DO_{10...0}$ memorijskog modula TAG0 i signali $\mathbf{TAG1_{11...0}}$ na izlaznim linijama podataka $DO_{10...0}$ memorijskog modula TAG1 i predstavljaju brojeve grupa blokova memorije

MEM koji se nalaze u memorijskom modulu TAG0 ulaza nula i memorijskom modulu TAG1 ulaza jedan seta određenog signalima $\mathbf{SET_{1..0}}$. Signali $\mathbf{TAG0_{11...0}}$ i $\mathbf{TAG1_{11...0}}$ vode na ulaze A komparatora CMP1 i CMP2. Na ulaze B komparatora CMP1 i CMP2 se dovode signali $\mathbf{CAR_{15..4}}$ generisane adrese koji predstavljaju broj bloka sa kojim treba realizovati operaciju čitanja, upisa ili selektivnog vraćanja. Signal jednakosti sa izlaza komparatora CMP0 zajedno sa signalom **V0**, koji određuje da li je blok u ulazu nula datog seta važeći, formiraju signal saglasnosti **H/M0** za ulaz nula datog seta. Signal jednakosti sa izlaza komparatora CMP1 zajedno sa signalom **V1**, koji određuje da li je blok u ulazu jedan datog seta važeći, formiraju signal saglasnosti **H/M1** za ulaz jedan datog seta. Signali saglasnosti ulaza nula **H/M0** i ulaza jedan **H/M1** formiraju signal saglasnosti **H/M** keš memorije ***KEŠ***.

Signali $\mathbf{CAR_{15...4}}$ se zbog operacija čitanja i upisa vode i na ulazne linije podataka $DI_{11...0}$ memorijskog modula TAG0 ulaza nula i memorijskog modula TAG1 radi upisa u jedan od ova dva modula broja grupe bloka memorije ***MEM*** koji se dovlači u keš memoriju ***KEŠ***. Adresa memorijskog modula u koji se vrši upis je određena signalima $\mathbf{SET_{1..0}}$ koji se vode na ulazne linije $A_{1...0}$ memorijskih modula TAG0 i TAG1 i predstavljaju broj seta u keš memoriji ***KEŠ***. Upis u jedan od ova dva memorijska modula se vrši generisanjem aktivne vrednosti signala **writeTAG**. Dekoder DC služi za formiranje signala **writeTAG0** i **writeTAG1**. Pri aktivnoj vrednosti signala **writeTAG**, a u zavisnosti od vrednosti signala **selSENT**, generiše se aktivna vrednost signala **writeTAG0** ili **writeTAG1**.

Multiplekser MP1, registar $TAGBUF_{11...0}$ i multiplekser MP2 u slučaju operacija čitanja, upisa, selektivnog vraćanja i kompletnog vraćanja formiraju signale $\mathbf{TAGOUT_{11...0}}$ koji predstavljaju broj grupe bloka podataka koji se vraća iz keš memorije ***KEŠ*** u memoriju ***MEM*** i koriste se u bloku *meminterfejs* radi formiranja adrese memorije ***MEM***.

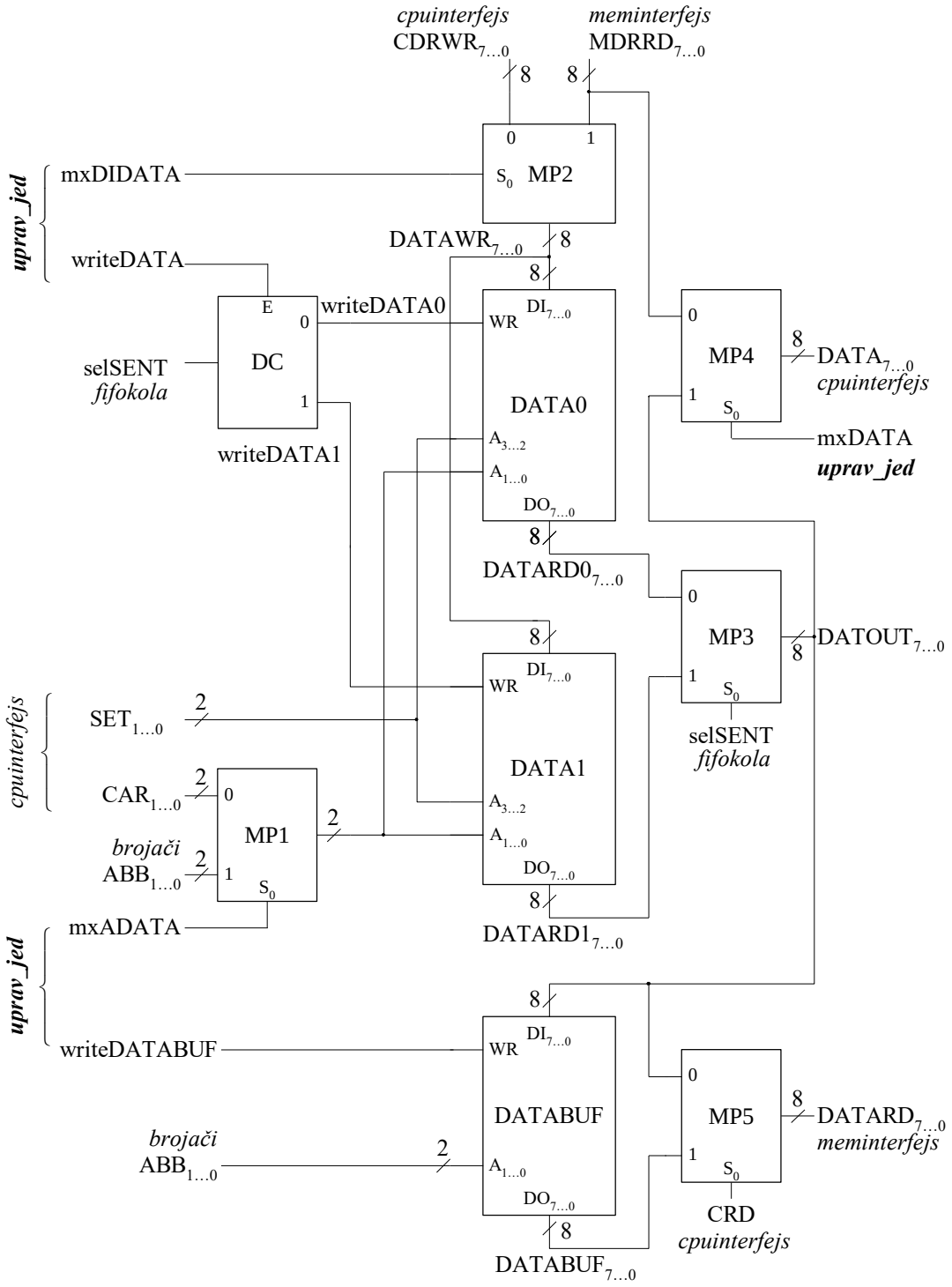
Multiplekser MP1 na svojim izlazima $\mathbf{TAG_{11...0}}$ propušta signale $\mathbf{TAG0_{11...0}}$ ili $\mathbf{TAG1_{11...0}}$. Selekcija se realizuje signalom **selSENT** u zavisnosti od toga da li se za dati set blok podataka vraća iz ulaza nula ili ulaza jedan.

Registar $TAGBUF_{11...0}$ služi za čuvanje broja grupe bloka podataka koji se u slučaju operacije čitanja kada treba blok podataka vratiti iz keš memorije ***KEŠ*** u memoriju ***MEM*** privremeno smešta u bafer podataka. Broj grupe očitan iz memorijskog modula TAG0 ili TAG1 se na ulaz registra $TAGBUF_{11...0}$ dovodi po linijama $\mathbf{TAG_{11...0}}$, a u registar $TAGBUF_{11...0}$ upisuje na signal takta pri aktivnoj vrednosti signala **ldTAGBUF**.

Kroz multiplekser MP2 se kod vraćanja bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM*** propušta sadržaj registra $TAGBUF_{11...0}$ ako je u toku operacija čitanja, odnosno vrednost sa linija $TAG_{11...0}$ ukoliko je u toku operacija upisa, selektivnog vraćanja ili kompletnog vraćanja. Selekcija sadržaja na linije $\mathbf{TAGOUT_{11...0}}$ se realizuje signalom **CRD**.

1.2.3.3.1.5. Blok datamem

Blok *datamem* (slika 75) sadrži memorijske module DATA0 i DATA1, multipleksere MP1, MP2, MP3 i MP4, dekoder DC, memorijski modul DATABUF i multiplekser MP5.



Slika 75 Blok datamem

Memorijski moduli DATA0 i DATA1 čuvaju sadržaje po četiri bloka koji se trenutno nalaze u keš memoriji. Svaki modul ima po 16 lokacija širine jedan bajt organizovanih u 4 bloka po 4 bajta. U modulima DATA0 i DATA1 se čuvaju blokovi iz nultih i prvih ulaza četiri seta keš memorije, respektivno.

Memorijski moduli DATA0 i DATA1 imaju sledeće ulaze i izlaze:

- adresne linije $A_{3...0}$,
- ulazne linije podataka $DI_{7...0}$,

- izlazne linije podataka $DO_{7...0}$ i
- upravljačku liniju WR.

Na adresne linije $A_{3..2}$ se dovode signali **$SET_{1..0}$** koji određuju broj seta. U slučaju operacija čitanja, upisa i selektivnog vraćanja signali **$SET_{1..0}$** se formiraju od signala **$CAR_{3...2}$** generisane adrese. U slučaju operacije kompletnog vraćanja signali **$SET_{1..0}$** se formiraju od signala **$CNTSET_{1...0}$** brojača $CNTSET_{1..0}$. Na adresne linije $A_{1..0}$ se dovode signali sa izlaza multipleksera MP1 koji određuju adresu bajta u bloku. Na ulaze multipleksera se dovode signali **$CAR_{1...0}$** generisane adrese i **$ABB_{1...0}$** brojača adresa u bloku $ABB_{1...0}$. Kroz multiplekser se signalom **mxADATA** propuštaju signali **$CAR_{1...0}$** kada je za operacije čitanja ili upisa otkrivena saglasnost i signali **$ABB_{1...0}$** kada se za operacije čitanja, upisa, selektivnog vraćanja i kompletnog vraćanja vrši prebacivanje bloka podataka između keš memorije ***KEŠ*** i memoriju ***MEM***.

Na ulazne linije podataka $DI_{7...0}$ se dovode signali **$DATAWR_{7...0}$** sa izlaza multipleksera MP2. Na ulaze multipleksera se dovode signali **$CDRWR_{7...0}$** koji predstavljaju zadati bajt podatka koji treba upisati u keš memoriju i signali **$MDRRD_{7...0}$** koji predstavljaju očitani bajt podatka iz memorije koji treba upisati u keš memoriju. Kroz multiplekser se signalom **mxDIDATA** propuštaju signali **$CDRWR_{7...0}$** kada je za operaciju upisa otkrivena saglasnost i signali **$MDRRD_{7...0}$** kada se vrši prebacivanje bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ***.

Na izlaznim linijama podataka **$DATARD0_{7...0}$** i **$DATARD1_{7...0}$** pojavljuje se bajtovi podataka očitani iz memorijskog modula DATA0 ulaza nula i memorijskog modula DATA1 ulaza nula. Multiplekser MP3 na svojim izlazima **$DATOUT_{7...0}$** propušta signale **$DATARD0_{7...0}$** ili **$DATARD1_{7...0}$**. Selekcija se realizuje signalom **selSENT** u zavisnosti od toga da li se za dati set bajt podatka čita iz ulaza nula ili ulaza jedan.

Upis u memorijske module DATA0 i DATA1 se realizuje aktivnom vrednošću signala **writeDATA0** odnosno **writeDATA1**, respektivno, a čitanje neaktivnim vrednostima ovih signala. Dekoder DC služi za formiranje signala **writeDATA0** i **writeDATA1**. Pri aktivnoj vrednosti signala **writeDATA**, a u zavisnosti od vrednosti signala **selSENT**, generiše se aktivna vrednost signala **writeDATA0** ili **writeDATA1**.

Multiplekser MP4 služi za selekciju signala **$DATAOUT_{7...0}$** ili **$MDRRD_{7...0}$** i njihovo prosleđivanje po linijama **$DATA_{7...0}$** u blok *cpuinterfejs* prilikom izvršavanja operacije čitanja. Ukoliko je u keš memoriji ***KEŠ*** otkrivena saglasnost podatak očitan iz memorijskog modula DATA0 ili DATA1 se pojavljuje na linijama **$DATAOUT_{7...0}$** i aktivnom vrednošću signala **mxDATA** propušta kroz multiplekser MP4 na linije **$DATA_{7...0}$**. Ukoliko u keš memoriji ***KEŠ*** nije otkrivena saglasnost dovlači se blok podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** i to počev od bajta bloka za koji je generisana adresa. Prvi bajt bloka očitan iz memorije ***MEM*** se pojavljuje na linijama **$MDRRD_{7...0}$** i aktivnom vrednošću signala **mxDATA** propušta kroz multiplekser multiplekser MP4 na linije **$DATA_{7...0}$**.

Memorijski modul DATABUF služi da se u njemi privremeno smesti bloka podataka koji u slučaju operacije čitanja treba da se vrati iz keš memorije ***KEŠ*** u memoriju ***MEM***. Bajtovi bloka podataka očitani iz memorijskih modula DATA0 ili DATA1 se dovode po linijama **$DATAOUT_{7...0}$** na ulazne linije podataka $DI_{7...0}$ memorijskog modula DATABUF, a adresa po linijama **$ABB_{1...0}$** na adresne linije $A_{1...0}$. Upis se realizuje pri aktivnoj vrednosti signala **writeDATABUF** koji se vodi na ulaz WR.

Kroz multiplekser MP5 se kod vraćanja bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM*** propušta sadržaj **$DATABUF_{7...0}$** sa izlaznih linija podataka $DO_{7...0}$ memorijskog modula DATABUF ako je u toku operacija čitanja, odnosno vrednost sa linija

DATAOUT$_{7...0}$ ukoliko je u toku operacija upisa, selektivnog vraćanja ili kompletnog vraćanja. Selekcija sadržaja na linije **DATARD$_{7...0}$** koje se vode u blok *meminterfejs* se realizuje signalom **CRD**.

Memorijskim modulima DATA0 i DATA1 pristupa se u sledećim slučajevima:

*1. Kod operacije čitanja pri čitanju bajta podatka iz keš memorije **KEŠ** ako je otkrivena saglasnost*. U ovom slučaju signali na adresnim linijama $A_{3...0}$ memorijskih modula DATA0 i DATA1 se formiraju od signala **CAR$_{3...2}$** i **CAR$_{1...0}$** sa izlaza registra $CAR_{15...0}$ bloka *cpuinterfejs*. Signali **CAR$_{3...2}$** se propuštaju kroz multiplekser MP sa slike 69 i pojavljuju kao signali **SET$_{1...0}$** na adresnim linijama $A_{3...2}$, dok se signali **CAR$_{1...0}$** propuštaju kroz multiplekser MP1 sa slike 75 i pojavljuju na adresnim linijama $A_{1...0}$. Očitani podaci se pojavljuju kao signali **DATARD0$_{7...0}$** i **DATARD1$_{7...0}$** na izlaznim linijama podataka $DO_{7...0}$ memorijskih modula DATA0 i DATA1, respektivno. Kroz multiplekser MP3 na linije **DATOUT$_{7...0}$** prolaze signali **DATARD0$_{7...0}$** ili **DATARD1$_{7...0}$**, jer u zavisnosti od toga da li je za dati set otkrivena saglasnost u ulazu nula ili ulazu jedan keš memorije signal **selSENT** ima neaktivnu ili aktivnu vrednost, respektivno. Signali **DATOUT$_{7...0}$** prolaze kroz multiplekser MP4 i po linijama **DATA$_{7...0}$** se vode u blok *cpuinterfejs*.

*2 Kod operacije upisa pri upisu bajta podatka u keš memoriju **KEŠ** ako je otkrivena saglasnost*. Adresa se formira kao u slučaju 1. Podatak za upis u memorijski modul DATA0 ili DATA1 se formira od signala **CDRWR$_{7...0}$** iz bloka *cpuinterfejs* koji se propuštaju kroz multiplekser MP2 i pojavljuju kao signali **DATAWR$_{7...0}$** na ulaznim linijama podataka $DI_{7...0}$ memorijskih modula DATA0 i DATA1. U zavisnosti od toga da li je za dati set otkrivena saglasnost u ulazu nula ili ulazu jedan keš memorije signal **selSENT** ima neaktivnu ili aktivnu vrednost, pa se aktivna vrednost signala **writeDATA** se pojavljuje kao aktivna vrednost signala **writeDATA0** ili **writeDATA1** i upis se realizuje u memorijski modul DATA0 ili DATA1.

*3. Kod operacija čitanja i upisa i to pri čitanju iz keš memorije **KEŠ** bajtova bloka podataka koji se vraćaju iz keš memorije **KEŠ** u memoriju **MEM**, jer nema saglasnosti za dati set a sadržaj bloka ulaza nula ili ulaza jedan seta odabranog za zamenu je modifikovan*. U ovom slučaju signali na adresnim linijama $A_{3...0}$ memorijskih modula DATA0 i DATA1 se formiraju od signala **CAR$_{3...2}$** i **ABB$_{1...0}$** sa izlaza registra $CAR_{15...0}$ bloka *cpuinterfejs* i brojača $ABB_{1...0}$ bloka *brojači*, respektivno. Signali **CAR$_{3...2}$** se propuštaju kroz multiplekser MP sa slike 69 i pojavljuju kao signali **SET$_{1...0}$** na adresnim linijama $A_{3...2}$, dok se signali **ABB$_{1...0}$** propuštaju kroz multiplekser MP1 sa slike 75 i pojavljuju na adresnim linijama $A_{1...0}$. Očitani podaci se pojavljuju kao signali **DATARD0$_{7...0}$** i **DATARD1$_{7...0}$** na izlaznim linijama podataka $DO_{7...0}$ memorijskih modula DATA0 i DATA1, respektivno. Kroz multiplekser MP3 na linije **DATOUT$_{7...0}$** prolaze signali **DATARD0$_{7...0}$** ili **DATARD1$_{7...0}$**, jer u zavisnosti od toga da li je za dati set za zamenu odabran blok iz ulaza nula ili ulaza jedan keš memorije signal **selSENT** ima neaktivnu ili aktivnu vrednost, respektivno. Sadržaj sa linija **DATOUT$_{7...0}$** se u slučaju operacije čitanja upisuje u memorijski modul DATABUF, a u slučaju operacije upisa propušta kroz multiplekser MP5 na linije **DATARD$_{7...0}$** i vodi u blok *meminterfejs*.

4. *Kod operacija čitanja i upisa i to pri upisu u keš memoriju **KEŠ** bajtova bloka podataka koji se dovlače iz memorije **MEM** u keš memoriju **KEŠ**, jer nema saglasnosti ni u ulazu nula ni u ulazu jedan datog seta*. Adresa se formira kao u slučaju 3. Podatak za upis u memorijski modul DATA0 ili DATA1 se formira od signala **MDRRD$_{7...0}$** iz bloka *meminterfejs* koji se propuštaju kroz multiplekser MP2 i pojavljuju kao signali **DATAWR$_{7...0}$** na ulaznim linijama podataka $DI_{7...0}$ memorijskih modula DATA0 i DATA1. U zavisnosti od toga da li je za dati set za zamenu odabran ulaz nula ili ulaz jedan keš memorije signal **selSENT** ima neaktivnu ili aktivnu vrednost, pa se aktivna vrednost signala **writeDATA** pojavljuje kao aktivna vrednost

signala **writeDATA0** ili **writeDATA1** i upis se realizuje u memorijski modul DATA0 ili DATA1.

5. *Kod operacije selektivnog vraćanja i to pri čitanju bajtova bloka koji se vraćaju iz keš memorije **KEŠ** u memoriju **MEM**, jer ima saglasnosti za dati set a sadržaj bloka ulaza nula ili ulaza jedan datog seta za koji je otkrivena saglasnost je modifikovan*. Formiranje adrese i čitanje bajtova bloka iz memorijskog modula DATA0 ili DATA1, propuštanje kroz multipleksere MP3 i MP5 i slanje u blok *meminterfejs* se realizuje kao u slučaju 3 za operaciju upisa.

6. *Kod operacije kompletnog vraćanja pri čitanju bajtova bloka podataka koji se vraćaju iz keš memorije **KEŠ** u memoriju **MEM**, jer je sadržaj bloka adresiranog ulaza i seta keš memorije **KEŠ** modifikovan*. U ovom slučaju signali na adresnim linijama $A_{3...0}$ memorijskih modula DATA0 i DATA1 se formiraju od signala $\mathbf{CNTSET_{1...0}}$ i $\mathbf{ABB_{1...0}}$ sa izlaza brojača $CNTSET_{1...0}$ i $ABB_{1...0}$ bloka *brojači*. Signali $\mathbf{CNTSET_{1...0}}$ se propuštaju kroz multiplekser MP bloka *cpuinterfejs* sa slike 69 i pojavljuju kao signali $\mathbf{SET_{1...0}}$ na adresnim linijama $A_{3...2}$, dok se signali $\mathbf{ABB_{1...0}}$ propuštaju kroz multiplekser MP1 sa slike 75 i pojavljuju na adresnim linijama $A_{1...0}$. Čitanje bajtova bloka iz memorijskog modula DATA0 ili DATA1, propuštanje kroz multipleksere MP3 i MP5 i slanje u blok *meminterfejs* se realizuje kao u slučaju 5.

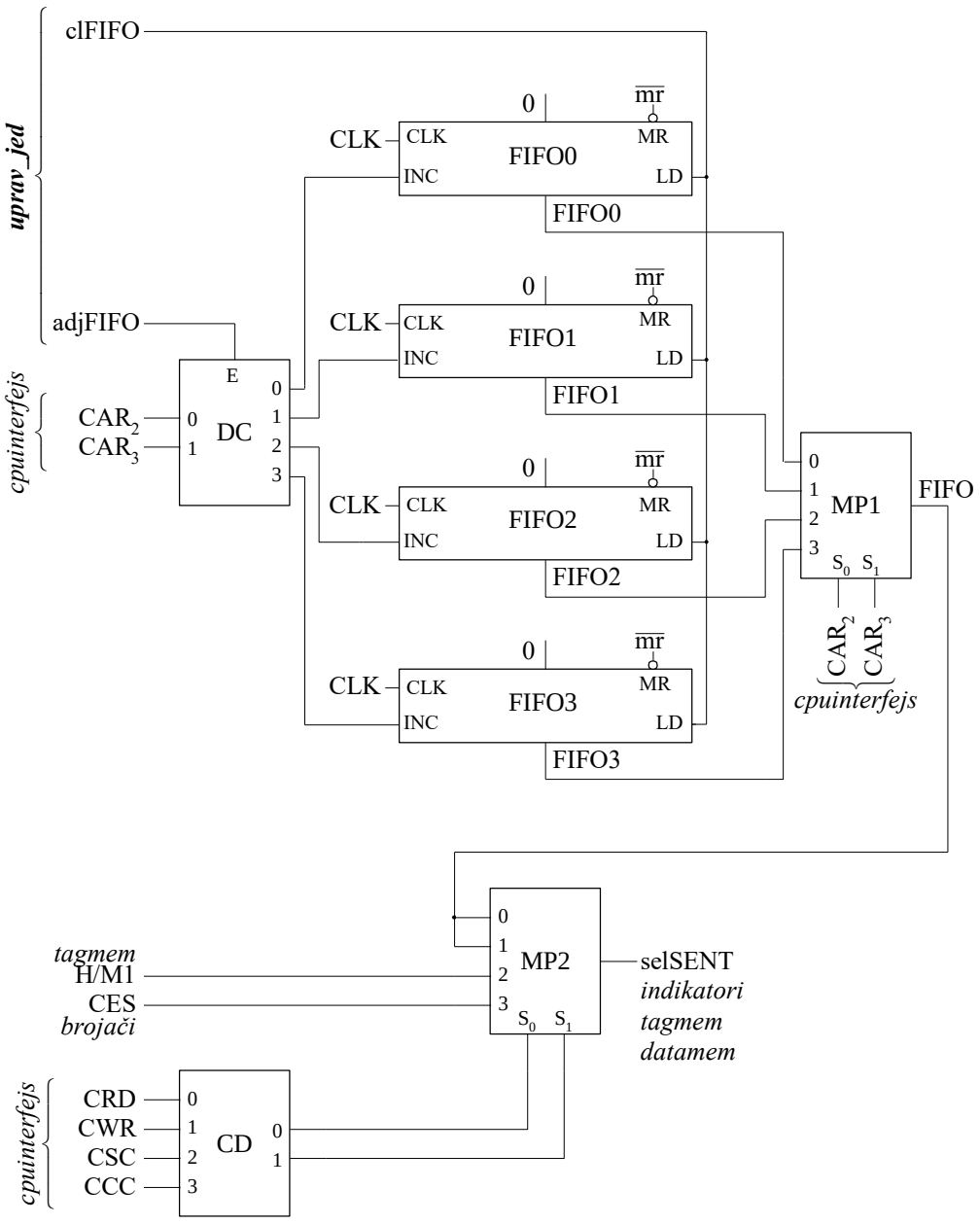
1.2.3.3.1.6. Blok fifokola

Blok *fifokola* (slika 76) sadrži 4 brojača FIFO0 do FIFO3, multipleksere MP1 i MP2, dekoder DC i koder CD.

Brojači FIFO0 do FIFO3 se koriste za realizaciju FIFO algoritma zamene blokova za keš memoriju sa 4 seta sa po dva ulaza po setu. Brojači FIFO0 do FIFO3 su jednorazredni, pri čemu u slučaju da nema saglasnosti ni u ulazu nula ni u ulazu jedan datog seta vrednost nula brojača određenog seta određuje da blok iz ulaza nula datog seta treba zameniti i brojač inkrementirati na jedan, dok vrednost jedan brojača određenog seta određuje da blok iz ulaza jedan datog seta treba zameniti i brojač inkrementirati na nulu. Na početku su svi brojači dovedeni na vrednost nula, a kasnije u toku rada njihove vrednosti se menjaju. Kada se u nekon trenutku utvrdi da u datom setu nema saglasnosti, na osnovu vrednosti brojača datog seta se utvrđuje da li iz ulaza nula ili ulaza jedan datog seta blok treba zameniti i generiše se aktivna vrednost signala **adjFIFO**. U zavisnosti od vrednosti signala $\mathbf{SET_{1...0}}$ na ulazima kodera CD, koji određuju kom setu se pristupa, aktivna vrednost signala **adjFIFO** se pojavljuje kao aktivna vrednost jednog od signala na izlazima kodera CD. Ovi signali se vode na ulaze INC brojača FIFO0 do FIFO3, pa aktivna vrednost jednog od ovih signala izaziva inkrementiranje jednog od brojača. Izlazi sva četiri brojača se vode na ulaze multipleksera MP1. U zavisnosti od vrednosti signala $\mathbf{SET_{1...0}}$ na ulazima multipleksera MP1, koji određuju kom setu se pristupa, na izlazu multipleksera MP1 se kao signal **FIFO** pojavljuje sadržaj brojača datog seta. Na kraju operacije kompletno vraćanje generiše se aktivna vrednost signala **clFIFO**. Ovaj signal se vodi na LD ulaze sva četiri FIFO brojača i njegovom aktivnom vrednošću se u ove brojače upisuje nula.

Multiplekser MP2 i koder CD se koriste za formiranje signala **selSENT** (*select Set ENTry*) koji vrednostima 0 i 1 određuje da li za dati set treba pristupiti bloku iz ulaza nula ili ulaza jedan keš memorije, respektivno. Signal **selSENT** na izlazu multipleksera MP2 dobija vrednost jednog od signala sa ulaza i to signala **FIFO** u slučaju operacije čitanja ili upisa, signala **H/M1** u slučaju operacije selektivno vraćanje ili signala **CES** u slučaju operacije kompletno vraćanje. Selekcija jednog od signala sa ulaza multipleksera MP2 se realizuje signalima koji dolaze sa izlaza kodera CD. Ovi signali se formiraju na osnovu signala **CRD**, **CWR**, **CSC** i **CCC** od kojih samo jedan može u datom trenutku da ima aktivnu vrednost u

zavisnosti od toga da li je u toku operacija čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja, respektivno.



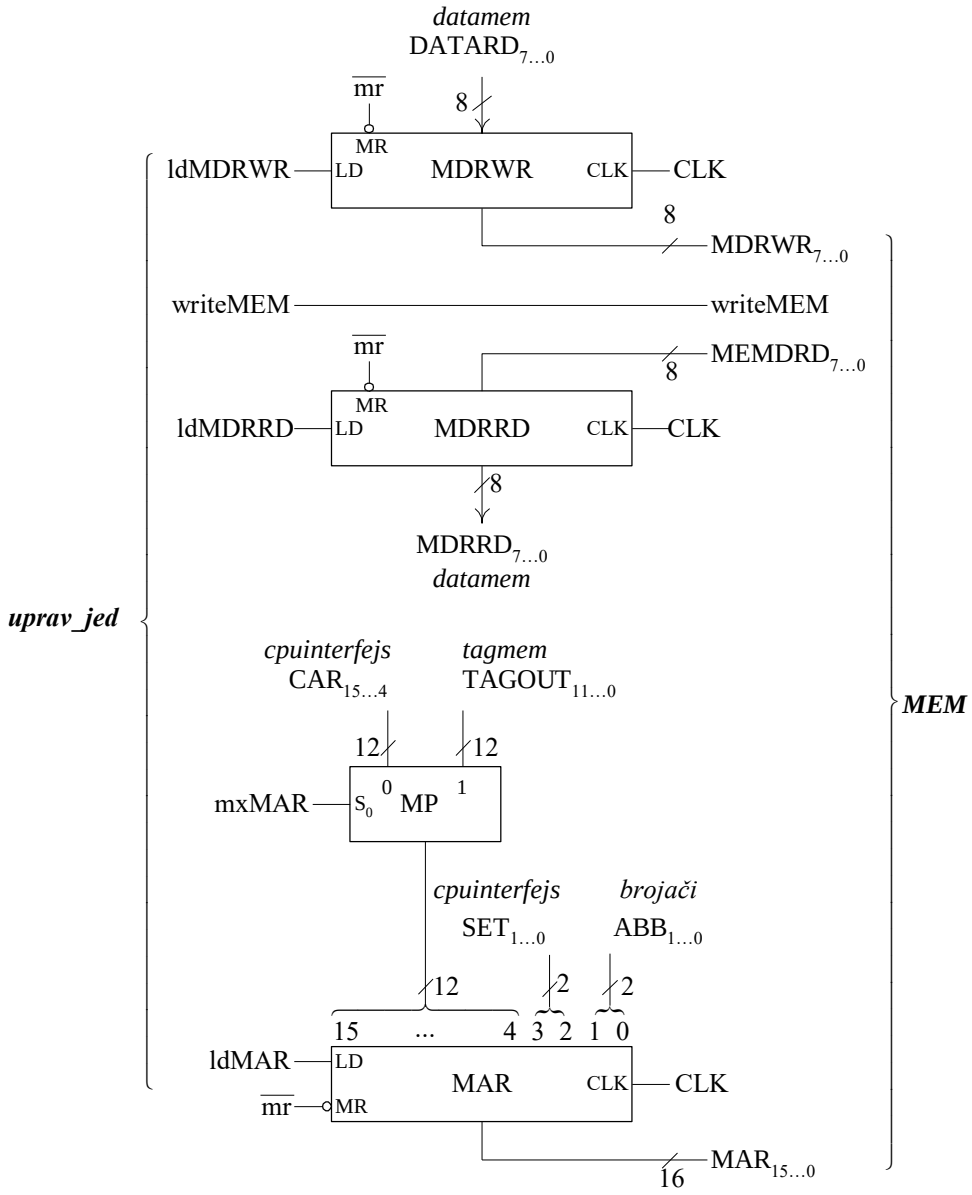
Slika 76 Blok fifokola

1.2.3.3.1.7. Blok meminterfejs

Blok *meminterfejs* (slika 77) sadrži registre MDRWR, MDRRD i MAR i multiplekser MP.

Registar $MDRWR_{7...0}$ (Memory Data Register for WRite) služi za čuvanje bajta podatka koji treba upisati u memoriju ***MEM*** pri vraćanju bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***. Na ulaze registra $MDRWR_{7...0}$ se iz bloka *datamem* po linijama $DATARD_{7...0}$ dovodi bajt podatka i to iz memorijskih modula DATA0 ili DATA1 kada je u toku operacija upisa, selektivnog vraćanja ili kompletnog vraćanja i iz memorijskog modula DATABUF kada je u toku operacija čitanja. Ovaj podatak se upisuje u registar $MDRWR_{7...0}$ na signal takta ukoliko je aktivan signal **ldMDRWR**. Sadržaj registra $MDRWR_{7...0}$ se vodi na

ulazne linije podataka memorije ***MEM***. Upis u memoriju ***MEM*** se realizuje pri aktivnoj vrednosti signala **writeMEM**.



Slika 77 Blok meminterfejs

Registar $MDRRD_{7...0}$ (Memory Data Register for ReaD) služi za čuvanje bajta podatka koji je očitan iz memorije ***MEM*** pri dovlačenju bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ***. Na ulaze registra $MDRRD_{7...0}$ dovodi se po linijama $MEMDRD_{7...0}$ bajt podatka sa izlaznih linija podataka memorije ***MEM***. Ovaj podatak se upisuje u registar $MDRRD_{7...0}$ na signal takta ukoliko je aktivan signal **ldMDRRD**. Izlazi registra $MDRRD_{7...0}$ se u slučaju operacija upisa i čitanja vode u blok *datamem* radi upisa u jedan od memorijskih modula DATA0 ili DATA1, a u slučaju operacije čitanja i u blok *cpuinterfejs* radi upisa u registar $CDRRD_{7...0}$.

Registar $MAR_{15...0}$ (Memory Address Register) služi za čuvanje ili adrese lokacije memorije ***MEM*** sa koje treba očitati podatak u slučaju dovlačenja bloka podataka ili adrese lokacije memorije ***MEM*** u koju treba upisati podatak u slučaju vraćanja bloka podataka.

Adresa se upisuje u registar $MAR_{15...0}$ na signal takta pri aktivnoj vrednosti signala **ldMAR**. Adresa koja se upisuje formira se iz tri dela na način dat u daljem tekstu.

- Kao bitovi 15 do 4 koristi se signali sa izlaza multipleksera MP i to $\mathbf{CAR_{15...4}}$ ili $\mathbf{TAGOUT_{11...0}}$. Signali $\mathbf{CAR_{15...4}}$ se propuštaju kroz multiplekser pri dovlačenju bloka podataka, a signali $\mathbf{TAGOUT_{11...0}}$ pri vraćanju bloka podataka. Pri tome se u slučaju operacija upisa, selektivnog vraćanja i kompletnog vraćanja signali $\mathbf{TAGOUT_{11...0}}$ formiraju na osnovu sadržaja iz memorijskih modula TAG0 ili TAG1 kada se podaci direktno vraćaju iz memorijskih modula DATA0 ili DATA1, dok se u slučaju operacije čitanja signali $\mathbf{TAGOUT_{11...0}}$ formiraju na osnovu sadržaja registra $TAGBUF_{11...0}$ kada se prethodno baferovani podaci vraćaju iz memorijskog modula DATABUF. Upravljačkim signalom **mxMAR** se kroz multiplekser MP selektuje jedna od te dve vrednosti. Signal **mxMAR** je, normalno, neaktivan, pa je tada na ulazima 15 do 4 registra $MAR_{15...0}$ prisutno $CAR_{15...4}$. Jedino kada treba propustiti $TAGOUT_{11...0}$ signal **mxMAR** dobija aktivnu vrednost.
- Kao bitovi 3 i 2 koriste se signali $\mathbf{SET_{1...0}}$ sa izlaza multipleksera MP bloka *cpuinterfejs* (slika 69). To su signali $\mathbf{CAR_{3...2}}$ registra $CAR_{15...0}$ iz bloka *cpuinterfejs* kada se pri operacijama čitanja i upisa blok podataka vraća i dovlači i kada se pri operaciji selektivnog vraćanja blok podataka vraća i signali $\mathbf{CNTSET_{1...0}}$ brojača $CNTSET_{1...0}$ iz bloka *brojači* kada se pri operaciji kompletnog vraćanja blok podataka vraća.
- Kao bitovi 1 i 0 koriste se signali $\mathbf{ABB_{1...0}}$ brojača bajtova bloka $ABB_{1...0}$ iz bloka *brojači*.

Izlazi registra $MAR_{15...0}$ se vode na adresne linije memorije ***MEM***.

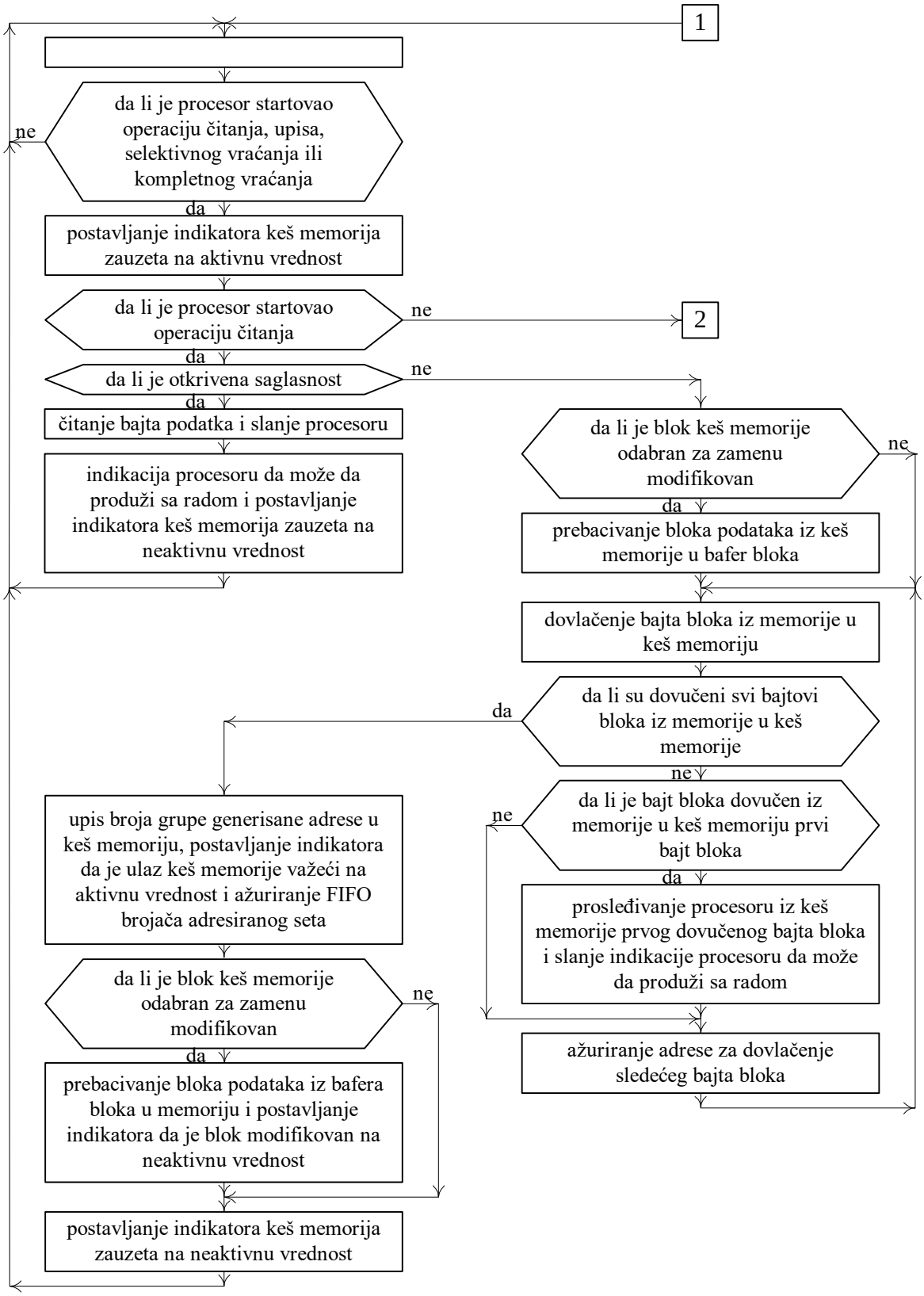
1.2.3.3.2. Upravljačka jedinica

U ovom poglavlju se daju dijagrami toka operacija, algoritam generisanja upravljačkih signala, vremenski oblici signala i struktura upravljačke jedinice.

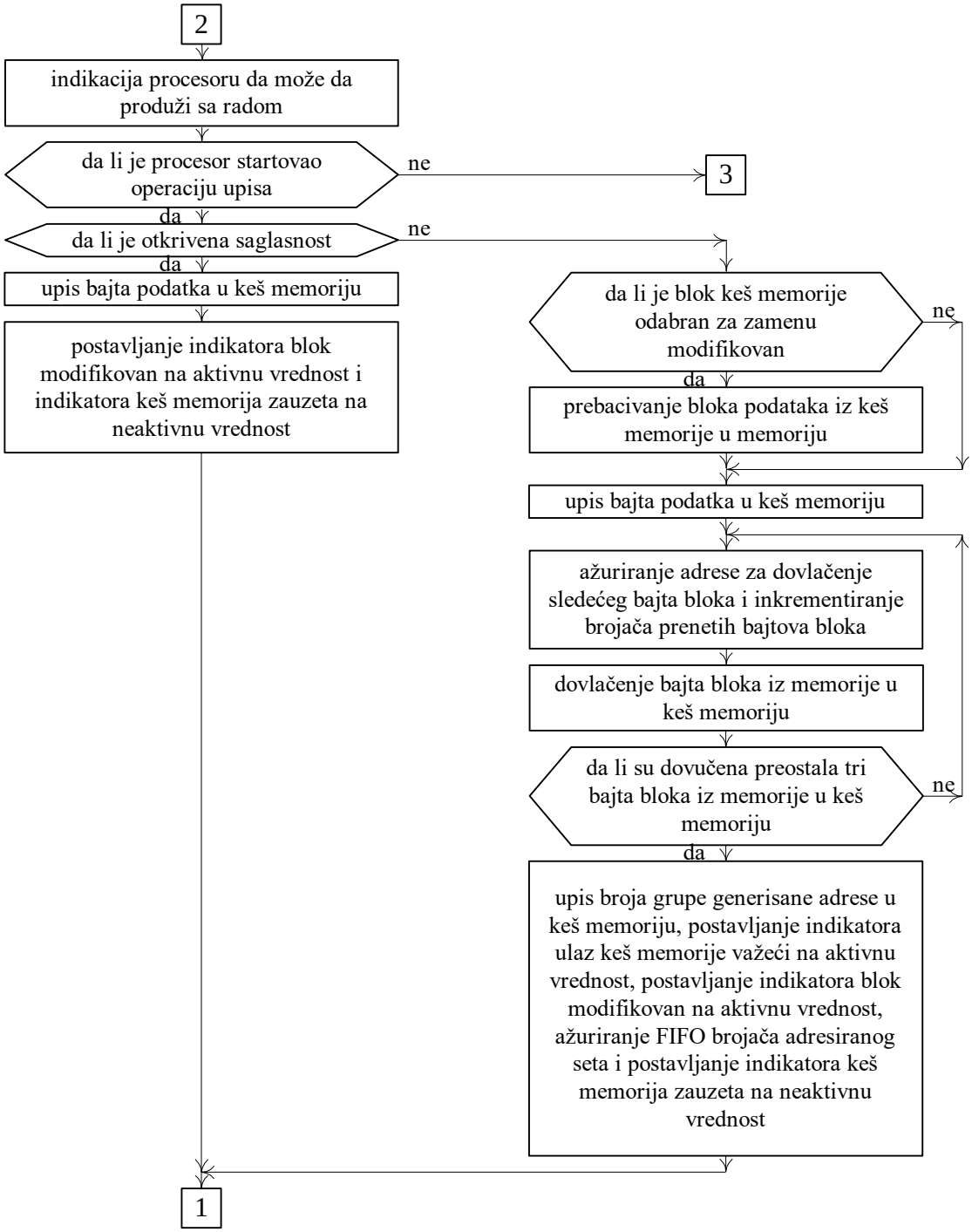
1.2.3.3.2.1. Dijagrami toka operacija

U ovom poglavlju se daju dijagrami toka operacija čitanja, upisa, selektivnog vraćanja i kompletnog vraćanja (78, 79, 80 i 81).

Upravljačka jedinica keš memorije čeka u početnom stanju pojavu signala zahteva za čitanje **PRQRD,** upis **PRQWR**, selektivno vraćanje **PRQSC** ili kompletno vraćanje **PRQCC** (slika 78). Pri pojavi jednog od prva tri zahteva, adresa koju šalje procesor se upisuje u prihvatni registar adrese keš memorije. Ukoliko je stigao zahtev za upis, podatak koji šalje procesor uz adresu se upisuje u prihvatni registar podatka keš memorije. Kada keš memorija prihvati zahtev od procesora, signal **BUSY** postaje aktivan i na taj način keš memorija signalizira procesoru da je zauzeta i da nije u mogućnosti da prihvati novi zahtev. U slučaju da zahtev nije za operaciju čitanja provera se vrši za operaciju upisa (slika 79). U slučaju da je zahtev za operaciju čitanja (slika 78) vrši se provera da li se sadržaj sa date adrese nalazi u keš memoriji. Ova provera će se dalje nazivati provera da li postoji saglasnost. Ukoliko postoji saglasnost vrši se čitanje podatka iz keš memorije i upis u prihvatni registar podatka keš memorije i procesoru šalje signal **CRP** kojim se signalizira procesoru da mu je traženi podatak na raspolaganju i da može da nastavi sa radom. Pored toga signal **BUSY** postaje neaktivan i na taj način keš memorija signalizira procesoru da više nije zauzeta i da može da prihvati novi zahtev. Na kraju se upravljačka jedinica keš memorije vraća u početno stanje i čeka novi zahtev za neku od operacija keš memorije (slika 78).



Slika 78 Dijagram toka operacije čitanja



Slika 79 Dijagram toka operacije upusa

Međutim, ukoliko ne postoji saglasnost treba da se pređe na dovlačenje bloka podataka iz memorije u keš memoriju. Ovome može da prethodi vraćanje bloka podataka iz keš memorije u memoriju ukoliko je sadržaj bloka odabranog za zamenu modifikovan, što je posledica odluke da se za ažuriranje memorije koristi metoda vrati-nazad (*write-back*). Da bi procesor došao do zahtevanog bajta podatka što je moguće pre, blok podataka se iz keš memorije privremeno smešta u bafer bloka i odmah se prelazi na dovlačenje bloka podataka iz memorije u keš memoriju i to počev od bajta bloka sa generisane adrese. Pri tome se prvi bajt bloka upisuje ne samo u keš memoriju već i u prihvatni registar podatka keš memorije i procesoru šalje signal **CRP** kojim se signalizira procesoru da mu je traženi podatak na raspolaganju i da može da nastavi sa radom, pa se posle toga iz memoriju u keš memoriju

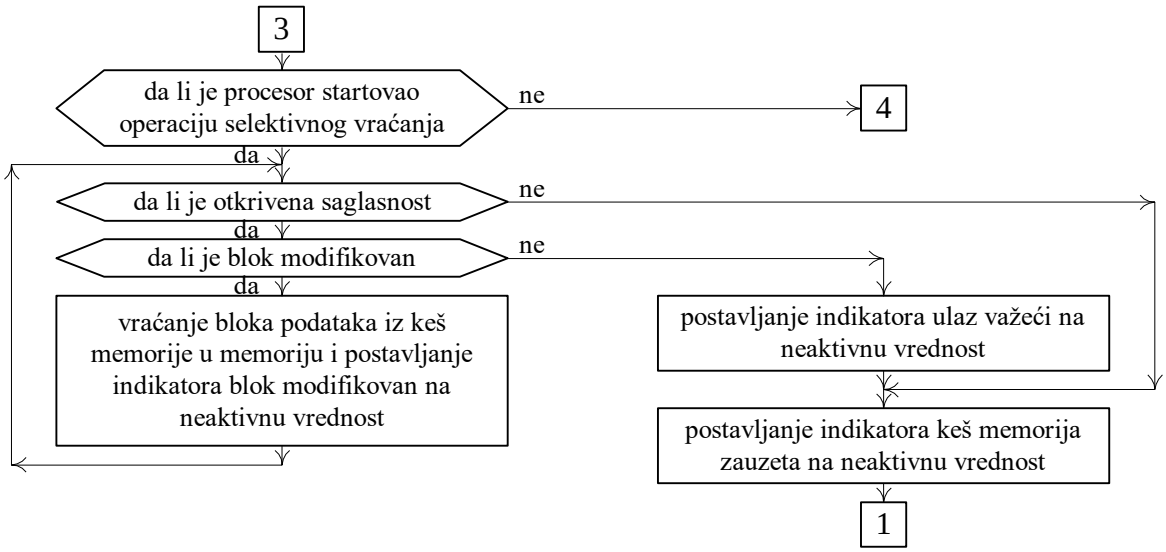
prebacuju i preostala tri bajta bloka podataka. Po završetku prebacivanja bloka podataka, u keš memoriju se upisuje i broj grupe datog bloka, indikator da je dati blok važeći postavlja na aktivnu vrednost i FIFO brojač seta kome pripada dati blok ažurira. Ukoliko je blok koji je odabran za zamenu modifikovan i zbog toga privremeno smešten u bafer bloka, sada se dati blok prebacuje iz bafera bloka u memoriju, a indikator blok modifikovan za ulaz keš memorije u kome se blok odabran za zamenu nalazio i u koji je već dovučen zahtevani blok podataka, postavlja na neaktivnu vrednost. Pored toga signal **BUSY** postaje neaktivan i na taj način keš memorija signalizira procesoru da više nije zauzeta i da može da prihvati novi zahtev. Na kraju se upravljačka jedinica keš memorije vraća u početno stanje i čeka novi zahtev za neku od operacija keš memorije (slika 78).

U slučaju da zahtev nije za operaciju čitanja (slika 78), već za operaciju upisa, selektivnog vraćanje ili kompletnog vraćanja, keš memorija šalje procesoru signal **CRP** kojim se signalizira procesoru da može da nastavi sa radom (slika 79). U slučaju da zahtev nije za operaciju upisa provera se vrši za operaciju selektivno vraćanje (slika 80). U slučaju da je zahtev za operaciju upisa (slika 79) vrši se provera da li u keš memoriji postoji saglasnost. Ukoliko postoji saglasnost, vrši se upis bajta podatka iz prihvatnog registra podatka u keš memoriju i indikator blok modifikovan postavlja na aktivnu vrednost. Pored toga signal **BUSY** postaje neaktivan i na taj način keš memorija signalizira procesoru da više nije zauzeta i da može da prihvati novi zahtev. Na kraju se upravljačka jedinica keš memorije vraća u početno stanje i čeka novi zahtev za neku od operacija keš memorije (slika 78).

Međutim, ukoliko ne postoji saglasnost treba da se pređe na dovlačenje bloka podataka iz memorije u keš memoriju, što je posledica odluke da se, kada pri operaciji upisa zahtevani blok nije u keš memoriji, koristi metoda njegovog dovlačenja i upisa samo u keš memoriju (*write-allocate*). Ovome može da prethodi vraćanje bloka podataka iz keš memorije u memoriju ukoliko je sadržaj bloka odabranog za zamenu modifikovan, što je posledica odluke da se za ažuriranje memorije koristi metoda vrati-nazad (*write-back*). Pri tome se prvi bajt bloka sa generisane adrese ne dovlači iz memorije u keš memoriju, već se u keš memoriju upisuje bajt podatka iz prihvatnog registra podatka, pa se posle toga iz memoriju u keš memoriju prebacuju i preostala tri bajta bloka podataka. Po završetku prebacivanja bloka podataka u keš memoriju se upisuje i broj grupe datog bloka, indikatori blok važeći i blok modifikovan, za ulaz keš memorije u kome se blok odabran za zamenu nalazio i u koji je dovučen zahtevani blok podataka, postavljaju na aktivnu vrednost i FIFO brojač seta kome pripada dati blok ažurira. Pored toga signal **BUSY** postaje neaktivan i na taj način keš memorija signalizira procesoru da više nije zauzeta i da može da prihvati novi zahtev. Na kraju se upravljačka jedinica keš memorije vraća u početno stanje i čeka novi zahtev za neku od operacija keš memorije (slika 78).

U slučaju da zahtev nije za operaciju upisa (slika 79) provera se vrši za operaciju selektivno vraćanje (slika 80). U slučaju da zahtev nije za operaciju selektivno vraćanje (slika 80), zahtev je za operaciju kompletno vraćanje (slika 81). U slučaju da je zahtev za operaciju selektivno vraćanje (slika 80), vrši se provera da li u keš memoriji postoji saglasnost. Ukoliko postoji saglasnost u nekom od dva ulaza adresiranog seta, vrši se provera da li je blok iz datog ulaza modifikovan. Ukoliko je blok modifikovan, onda se dati blok podataka vraća iz keš memorije u memoriju, indikator blok modifikovan postavlja na neaktivnu vrednost i ponovo vrši provera da li u keš memoriji postoji saglasnost. I ova provera daje da postoji saglasnost, pa se sada vrši provera da li je blok modifikovan. Međutim, ova provera sada daje da blok nije modifikovan. Ukoliko blok nije modifikovan, indikator ulaz važeći za ulaz keš memorije adresiranog seta za koji je otkrivena saglasnost se postavlja na neaktivnu vrednost. Pored toga signal **BUSY** postaje neaktivan i na taj način keš memorija signalizira procesoru da više nije

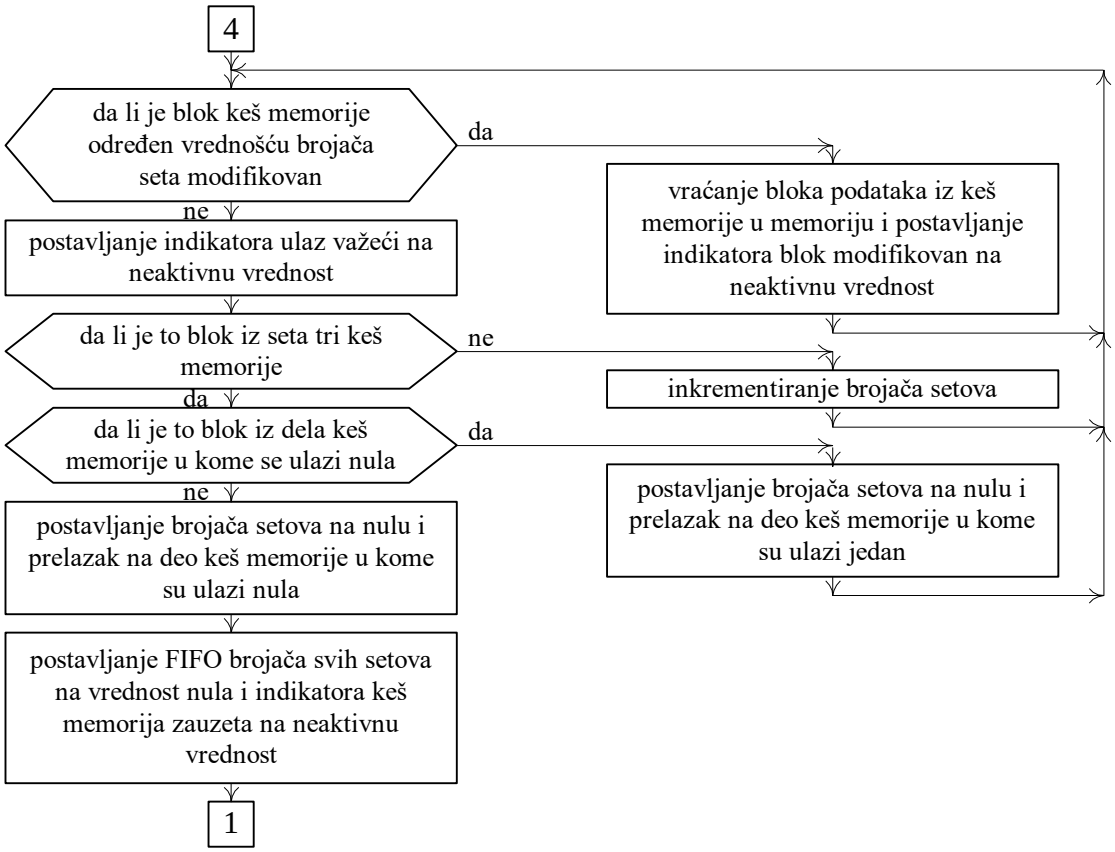
zauzeta i da može da prihvati novi zahtev. Na kraju se upravljačka jedinica keš memorije vraća u početno stanje i čeka novi zahtev za neku od operacija keš memorije (slika 78) .



Slika 80 Dijagram toka operacije selektivnog vraćanja

Međutim, ukoliko ne postoji saglasnost samo signal **BUSY** postaje neaktivan i na taj način keš memorija signalizira procesoru da više nije zauzeta i da može da prihvati novi zahtev. Na kraju se upravljačka jedinica keš memorije vraća u početno stanje i čeka novi zahtev za neku od operacija keš memorije (slika 78).

U slučaju da zahtev nije za operaciju selektivno vraćanje (slika 80), zahtev je za operaciju kompletno vraćanje (slika 81). Operacija se realizuje redom najpre za blokove iz ulaza nula za setove 0, 1, 2 i 3, a zatim za blokove iz ulaza jedan za setove 0, 1, 2 i 3. Ako prva provera za blok iz ulaza nula seta nula pokaže da je blok modifikovan, blok se vraća iz keš memorije u memoriju i indikator blok modifikovan postavlja na neaktivnu vrednost. Druga provera za isti blok pokazuje da blok sada nije modifikovan, pa se indikator ulaz važeći postavlja na neaktivnu vrednost. Ako prva provera za blok iz ulaza nula seta nula pokaže da blok nije modifikovan, indikator ulaz važeći se odmah postavlja na neaktivnu vrednost. Sada se vrši provera da li je dati blok bio iz ulaza nula seta tri. Ukoliko nije, vrši se samo inkrementiranje brojača setova i prethodni postupak se ponavlja za blokove iz ulaza nula setova 1, 2 i 3. Ukoliko jeste, brojač setova se postavlja na vrednost nula i prelazi na blokove iz ulaza jedan. Na isti način se operacija realizuje i za blokove iz ulaza jedan sva četiri seta. Kada se operacija realizuje za blok ulaza jedan seta tri, brojač setova se postavlja na vrednost nula i prelazi na blokove iz ulaza nula. Time je keš memorija dovedena u početno stanje za realizaciju sledeće operacije kompletno vraćanje. Posle toga se FIFO brojači sva četiri seta postavljaju na vrednost nula. Pored toga signal **BUSY** postaje neaktivan i na taj način keš memorija signalizira procesoru da više nije zauzeta i da može da prihvati novi zahtev. Na kraju se upravljačka jedinica keš memorije vraća u početno stanje i čeka novi zahtev za neku od operacija keš memorije (slika 78).



Slika 81 Dijagram toka operacije kompletnog vraćanja

1.2.3.3.2.2. Sekvenca upravljačkih signala po koracima

Sekvenca upravljačkih signala po koracima je formirana na osnovu strukture operacione jedinice (poglavlje 1.2.3.3.1) i dijagrama toka operacija (poglavlje 1.2.3.3.2.1). Notacija koja se koristi je identična sa notacijom iz poglavlja 1.2.1.1.2.2.

Sekvenca upravljačkih signala po koracima je data u daljem tekstu.

! U koraku $step_0$ se čeka da iz procesora preko bloka *cpuinterfejs* stigne jedan od signal startovanja operacije čitanja **PRQRD,** upisa **PRQWR**, selektivnog vraćanja **PRQSC** ili kompletnog vraćanja **PRQCC**. Ako se pojavi signal **PRQRD** na signal takta se adresa upisuje u registar CAR bloka *cpuinterfejs* i flip-flop CRD bloka *cpuinterfejs* postavlja na aktivnu vrednost. Ako se pojavi signal **PRQWR** na signal takta se adresa upisuje u registar CAR, podatak u registar CDRWR bloka *cpuinterfejs* i flip-flop CWR bloka *cpuinterfejs* postavlja na aktivnu vrednost. Ako se pojavi signal **PRQSC** na signal takta se adresa upisuje u registar CAR i flip-flop CSC bloka *cpuinterfejs* postavlja na aktivnu vrednost. Ako se pojavi signal **PRQCC** na signal takta flip-flop CCC bloka *cpuinterfejs* se postavlja na aktivnu vrednost. Postavljanjem jednog od flip-flopova CRD, CWR, CSC ili CCC na aktivnu vrednost i signal zauzetosti keš memorije **BUSY** bloka *cpuinterfejs* postaje aktivan. Ako se pojavi bilo koji od signala **PRQRD, PRQWR, PRQSCL** ili **PRQCC** na signal takta se prelazi na korak $step_1$. U suprotnom slučaju se ostaje u koraku $step_0$. !

$step_0$: *br* (*if* (**PRQRD** + **PRQWR** + **PRQSC** + **PRQCC**) *then* $step_1$ *else* $step_0$),

! U korak $step_1$ može da se dođe iz koraka $step_0$ i iz koraka $step_4$. Iz koraka $step_0$ se dolazi uvek kada se startuje operacija čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja. Iz koraka $step_4$ se dolazi u slučaju operacije selektivno vraćanje ili kompletno vraćanje po izvršenom prebacivanju bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***.

U koraku $step_1$ se za operacije čitanja, upisa i selektivnog vraćanja vrši provera saglasnosti i na osnovu toga formira vrednost signala **H/M** bloka *tagmem*. Aktivna vrednost signala **H/M** označava da je otkrivena saglasnost. U koraku $step_1$ se za sve četiri operacije pod određenim uslovima vrši provera da li je sadržaj bloka

podataka određenog ulaza keš memorije modifikovan i na osnovu toga formira vrednost signala **D** bloka *indikatori*. Aktivna vrednost signala **D** označava da je sadržaj modifikovan. U koraku $step_1$ se za operaciju kompletnog vraćanja pod određenim uslovima vrši i provera vrednosti signala **CLEAN** bloka *brojači*. Aktivna vrednost signala **CLEAN** je indikacija da su svi modifikovan blokovi keš memorije ***KEŠ*** vraćeni u memoriju ***MEM***. Signal **CLEAN** postaje aktivan pri aktivnoj vrednosti signala **indCNTSET** trajanja jedna perioda signala takta ako brojač CNTSET bloka *brojači* inkrementiranjem od početne vrednosti nula dostigne krajnju vrednost tri i pri tome je flip-flop CES postavljen na aktivnu vrednost.

U slučaju operacije čitanja signal **CRD** je aktivan. Ako je formirana neaktivna vrednost signala **H/M** generiše se aktivna vrednost signala **ldABB** bloka *brojači*, pa se na signal takta adresa bajta u bloku generisane adrese upisuje u brojač ABB bloka *brojači*. Ako je formirana aktivna vrednost signala **H/M** generiše se aktivna vrednost signala **ldCDRRD** bloka *cpuinterfejs*, pa se na signal takta vrednost očitana iz memorijskih modula DATA0 ili DATA1 bloka *datamem* upisuje u registar CDRRD bloka *cpuinterfejs*. U slučaju operacija čitanja iz koraka $step_1$ se prelazi na korak $step_2$, $step_8$ ili $step_5$ u zavisnosti od vrednosti signala **H/M** i **D**. Ako je signal **H/M** aktivan, prelazi se na korak $step_2$ radi kompletiranja date operacije. Ako je signal **H/M** neaktivan, prelazi se u zavisnosti od vrednosti signala **D** na korak $step_8$ ili korak $step_5$. Pri aktivnoj vrednosti signala **D**, što znači da je sadržaj bloka odabranog za zamenu modifikovan, prelazi se na korak $step_8$. U koraku $step_8$ se vrši prebacivanje bajtova bloka koji je odabran za zamenu i čiji je sadržaj modifikovan u bafer bloka. Pri neaktivnoj vrednosti signala **D**, što znači da sadržaj bloka odabranog za zamenu nije modifikovan, prelazi se na korak $step_5$. U koracima $step_5$, $step_6$ i $step_7$ se dovlače bajtovi bloka iz memorije ***MEM*** u keš memoriju ***KEŠ***.

U slučaju operacije upisa signal **CWR** je aktivan. U ovom koraku se bezuslovno generiše aktivna vrednost signal **CRP** bloka *cpuinterfejs* da bi se procesoru ***CPU*** signaliziralo da može da produži sa radom bez obzira na to što keš memorija ***KEŠ*** nije još uvek kompletirala operaciju upisa. Ako je formirana neaktivna vrednost signala **H/M** generiše se aktivna vrednost signala **ldABB** bloka *brojači*, pa se na signal takta adresa bajta u bloku generisane adrese upisuje u brojač ABB bloka *brojači*. U slučaju operacije upisa iz koraka $step_1$ se prelazi na korak $step_2$, $step_3$ ili $step_5$ u zavisnosti od vrednosti signala **H/M** i **D**. Ako je signal **H/M** aktivan, prelazi se na korak $step_2$ radi upisa bajta podatka u keš memoriju ***KEŠ*** i kompletiranja date operacije. Ako je signal **H/M** neaktivan, prelazi se u zavisnosti od vrednosti signala **D** na korak $step_3$ ili korak $step_5$. Pri aktivnoj vrednosti signala **D**, što znači da je sadržaj bloka odabranog za zamenu modifikovan, prelazi se na korak $step_3$. U koracima $step_3$ i $step_4$ se vraćaju bajtovi bloka iz keš memorije ***KEŠ*** u memoriju ***MEM***. Pri neaktivnoj vrednosti signala **D**, što znači da sadržaj bloka nije modifikovan, prelazi se na korak $step_5$. U koracima $step_5$, $step_6$ i $step_7$ se dovlače bajtovi bloka iz memorije ***MEM*** u keš memoriju ***KEŠ***.

U slučaju operacije selektivnog vraćanja signal **CSC** je aktivan. U ovom koraku se bezuslovno generiše aktivna vrednost signal **CRP** bloka *cpuinterfejs* da bi se procesoru ***CPU*** signaliziralo da može da produži sa radom bez obzira na to što keš memorija ***KEŠ*** nije još uvek kompletirala operaciju selektivnog vraćanja. Ako je formirana aktivna vrednost signala **H/M** i neaktivna vrednost signala **D** generiše se aktivna vrednost signala **clearV** bloka *indikatori*. Ovim signalom se flip-flop V odgovarajućeg ulaza keš memorije ***KEŠ*** postavlja na neaktivnu vrednost, čime se dati ulaz proglašava za nevažeći. Ako su formirane aktivne vrednosti signala **H/M** i **D** generiše se aktivna vrednost signala **ldABB** bloka *brojači*, pa se na signal takta adresa bajta u bloku generisane adrese upisuje u brojač ABB bloka *brojači*. Iz koraka $step_1$ se prelazi ili na korak $step_2$ ili na korak $step_3$. U korak $step_2$ se prelazi ukoliko je signal **H/M** neaktivan ili ukoliko je signal **H/M** aktivan a signal **D** neaktivan. U korak $step_2$ se prelazi radi kompletiranja operacije. U korak $step_3$ se prelazi ukoliko su signali **H/M** i **D** aktivni. U koracima $step_3$ i $step_4$ se vraćaju bajtovi bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***, flip-flop D odgovarajućeg ulaza keš memorije ***KEŠ*** postavlja na neaktivnu vrednost i ponovo vraća na korak $step_1$.

U slučaju operacije kompletnog vraćanja signal **CCC** je aktivan. U ovom koraku se bezuslovno generiše aktivna vrednost signal **CRP** bloka *cpuinterfejs* da bi se procesoru ***CPU*** signaliziralo da može da produži sa radom bez obzira na to što keš memorija ***KEŠ*** nije još uvek kompletirala operaciju kompletnog vraćanja. Ako je formirana neaktivna vrednost signala **D** generiše se aktivna vrednost signala **clearV** bloka *indikatori* i **incCNTCE** bloka *brojači*. Signalom **clearV** se flip-flop V odgovarajućeg ulaza keš memorije ***KEŠ*** postavlja na neaktivnu vrednost, čime se dati ulaz proglašava za nevažeći. Signalom **incCNTCE** bloka *brojači* se inkrementira sadržaj brojača CNTSET, čime se prelazi na sledeći ulaz keš memorija ***KEŠ***. Iz koraka $step_1$ se prelazi ili na korak $step_3$ ili na korak $step_2$ ili se ostaje u koraku $step_1$. U korak $step_3$ se prelazi ukoliko je signal **D** aktivan. U koracima $step_3$ i $step_4$ se vraćaju bajtovi bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***, flip-flop D odgovarajućeg ulaza keš memorije ***KEŠ*** postavlja na neaktivnu vrednost i ponovo vraća na korak $step_1$. U korak $step_2$ se prelazi ukoliko je signal **D** neaktivan, a signal **CLEAN** aktivan. U korak $step_2$ se prelazi radi kompletiranja operacije. U koraku $step_1$ se ostaje ukoliko su signali **D** i **CLEAN** neaktivni. !

$step_1$: *if* (**CRD**·$\overline{\text{H/M}}$ + **CWR**·$\overline{\text{H/M}}$ + **CSC**·**H/M**·**D**) *then* **ldABB**,
if (**CRD**·**H/M**) *then* **ldCDRRD**,
if (**CSC**·**H/M**·$\overline{\text{D}}$) *then* **clearV**,

if (**CCC**·$\overline{\mathbf{D}}$) *then* (**incCNTSET, clearV**),
if (**CWR** + **CSC** + **CCC**) *then* **CRP**,
br (*if* **CRD**
then (*if* **H/M** *then* $step_2$ *else* (*if* **D** *then* $step_8$ *else* $step_5$))
else (*if* **CWR**
then (*if* **H/M** *then* $step_2$ *else* (*if* **D** *then* $step_3$ *else* $step_5$))
else (*if* **CSC** *then* (*if* **H/M**
then (*if* **D** *then* $step_3$ *else* $step_2$)
else $step_2$)
else (*if* **D**
then $step_3$
else (*if* **CLEAN** *then* $step_2$ *else* $step_1$))))),

! U korak $step_2$ može da se dođe iz koraka $step_1$ i iz koraka $step_4$. U slučaju operacije čitanja dolazi se samo iz koraka $step_1$ i to samo ukoliko postoji saglasnost. U slučaju operacije upisa dolazi se iz koraka $step_1$ ukoliko postoji saglasnost i iz koraka $step_4$ ukoliko ne postoji saglasnost. U slučaju operacija selektivno vraćanje i kompletno vraćanje uvek se dolazi iz koraka $step_1$.

U slučaju operacije čitanja signal **CRD** je aktivan. Tada se generišu aktivna vrednost signala **CRP**, koji keš memorija ***KEŠ*** preko bloka *cpuinterfejs* šalje procesoru ***CPU,*** i signala **clCRD** bloka *cpuinterfejs*. Aktivna vrednost signala **CRP** je indikacija procesoru ***CPU*** da je bajt podatka pročitan iz keš memorije ***KEŠ*** i da procesor ***CPU*** može da ga prebaci u svoj prihvatni registar i produži sa radom. Aktivnom vrednošću signala **clCRD** se na signal takta upisuje neaktivna vrednost u flip-flop CRD, čime i signal zauzetosti keš memorije **BUSY** postaje neaktivan. Time je operacija čitanja kompletirana. Na signal takta se uvek prelazi iz koraka $step_2$ u korak $step_0$.

U slučaju operacije upisa signal **CWR** je aktivan. Ukoliko je aktivna vrednost signala **H/M** generiše se aktivna vrednost signala **writeDATA** bloka *datamem* kojim se upisuje bajt podatka u memorijski modul DATA0 ili DATA1. Pri tome se još generišu aktivne vrednosti signala **writeD** bloka *indikatori* i **clCWR** bloka *cpuinterfejs*, pa se na signal takta upisuje aktivna vrednost u odgovarajući flip-flop D i neaktivna vrednost u flip-flop CWR. Postavljanjem flip-flopa CWR na neaktivnu vrednost i signal zauzetosti keš memorije **BUSY** postaje neaktivan. Time je operacija upisa kompletirana. Na signal takta se prelazi iz koraka $step_2$ u korak $step_0$. Ukoliko je neaktivna vrednost signala **H/M** generišu se aktivne vrednosti signala **mxADATA** i **writeDATA** bloka *datamem* kojima se upisuje bajt podatka u memorijski modul DATA0 ili DATA1. Pri tome se još generišu aktivne vrednosti signala **incABB** i **incCNTBB** bloka *brojači*, pa se na signal takta inkrementiraju sadržaji brojača **ABB** i **CNTBB**. Time operacija upisa nije kompletirana. Pre prelaska iz koraka $step_4$ na korak $step_2$, najpre je u koraku $step_1$ utvrđeno da nema saglasnosti, pa je u koracima $step_3$ i $step_4$ izvršeno prebacivanje iz keš memorije ***KEŠ*** u memoriju ***MEM*** bajtova bloka odabranog za zamenu. U koraku $step_2$ je u dati blok keš memoriju ***KEŠ*** upisan bajt bloka sa adrese za koju je startovana operacija upisa, dok preostala tri bajta bloka treba da se dovuku iz memorije ***MEM*** u keš memoriju ***KEŠ*** u koracima $step_5$, $step_6$ i $step_7$. Stoga se na signal takta prelazi iz koraka $step_2$ u korak $step_5$.

U slučaju operacije selektivnog vraćanja signal **CSC** je aktivan. Tada se generiše aktivna vrednost signala **clCSC** bloka *cpuinterfejs*. Aktivnom vrednošću signala **clCSC** se na signal takta upisuje neaktivna vrednost u flip-flop CSC, čime i signal zauzetosti keš memorije **BUSY** postaje neaktivan. Time je operacija selektivnog vraćanja kompletirana. Na signal takta se uvek prelazi iz koraka $step_2$ u korak $step_0$.

U slučaju operacije kompletnog vraćanja signal **CCC** je aktivan. Tada se generišu aktivna vrednost signala **clearFIFO** bloka *fifokola* i signala **clCCC** bloka *cpuinterfejs*. Aktivnom vrednošću signala **clearFIFO** se na signal takta svi FIFO brojači postavljaju na početnu vrednost nula. Aktivnom vrednošću signala **clCCC** se na signal takta upisuje neaktivna vrednost u flip-flop CCC, čime i signal zauzetosti keš memorije **BUSY** postaje neaktivan. Time je operacija kompletnog vraćanja kompletirana. Na signal takta se uvek prelazi iz koraka $step_2$ u korak $step_0$. !

$step_2$: *if* **CRD** *then* (**CRP**, **clCRD**),
if (**CWR**·**H/M**) *then* (**writeDATA**, **writeD**, **clCWR**),
if (**CWR**·$\overline{\mathbf{H/M}}$) *then* (**mxADATA**, **writeDATA**, **incABB**, **incCNTBB**),
if **CSC** *then* **clCSC**,
if **CCC** *then* (**clearFIFO**, **clCCC**),
br (*if* (**CWR**·$\overline{\mathbf{H/M}}$) *then* $step_5$ *else* $step_0$),

! U korak $step_3$ se dolazi iz koraka $step_1$ i iz koraka $step_4$. Iz koraka $step_1$ se dolazi ukoliko se u slučaju operacije upisa u njemu otkrije da nema saglasnosti i da je blok keš memorije ***KEŠ*** odabran za zamenu modifikovan pa ga

treba vratiti iz keš memorije ***KEŠ*** u memoriju ***MEM***. Iz koraka $step_1$ se dolazi u slučaju operacije selektivnog vraćanja ukoliko se u njemu otkrije da postoji saglasnost i da je adresirani blok keš memorije ***KEŠ*** modifikovan pa ga treba vratiti iz keš memorije ***KEŠ*** u memoriju ***MEM***. Iz koraka $step_1$ se dolazi u slučaju operacije kompletnog vraćanja ukoliko se otkrije da je adresirani blok keš memorije ***KEŠ*** modifikovan pa ga treba vratiti iz keš memorije ***KEŠ*** u memoriju ***MEM***. Iz koraka $step_4$ se dolazi po prenosu jednog bajta bloka iz keš memorije ***KEŠ*** u memoriju ***MEM***.

U koraku $step_3$ se bezuslovno generišu aktivne vrednosti signala **mxADATA** bloka *datamem* i signala **ldMDRWR**, **mxMAR** i **ldMAR** bloka *meminterfejs*. Aktivnom vrednošću signala **mxADATA** se u bloku *datamem* kroz multiplekser propušta vrednost brojača ABB, dok se aktivnom vrednošću signala **ldMDRWR** obezbeđuje da se podatak $DATARD_{7...0}$ očitan iz memorijskog modula DATA0 ili DATA1 bloka *datamem* upiše na signal takta u registar MDRWR bloka *meminterfejs*. U bloku *meminterfejs* se aktivnom vrednošću signala **mxMAR** kroz multiplekser propušta vrednost $TAGOUT_{11...0}$ očitana iz memorijskog modula TAG0 ili TAG1 bloka *tagmem*, dok se aktivnom vrednošću signala **ldMAR** na signal takta u registar MAR upisuje formirana adresa.

Na signal takta se uvek prelazi iz koraka $step_3$ na korak $step_4$. !

$step_3$: **mxADATA**, **ldMDRWR**, **mxMAR**, **ldMAR**,
br $step_4$,

! U korak $step_4$ se dolazi samo iz koraka $step_3$ i to samo za operacije upisa, selektivnog vraćanja i kompletnog vraćanja ukoliko treba vratiti blok podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***.

U koraku $step_4$ se bezuslovno generišu aktivne vrednosti signala **writeMEM** bloka *meminterfejs* i **incMEMACC** bloka *brojači*. Aktivnom vrednošću signala **writeMEM** se sadržaj registra MDRWR bloka *meminterfejs* upisuje u memoriju ***MEM*** na adresi određenoj sadržajem registra MAR bloka *meminterfejs*. Aktivnom vrednošću signala **incMEMACC** se obezbeđuje da se pri pojavi signala takta inkrementira sadržaj brojača MEMACC koji određuje vreme pristupa memoriji ***MEM***. Pri aktivnoj vrednosti signala **MEMFC** bloka *brojači* generišu se aktivne vrednosti signala **incABB** i **incCNTBB** bloka *brojači*, pa se pri pojavi signala takta inkrementiraju brojači ABB i CNTBB.

U slučaju operacija selektivnog vraćanja i kompletnog vraćanja aktivni su signali **CSC** i **CCC** bloka *cpuinterfejs*, respektivno. Ukoliko su tada aktivne vrednosti signala **CNTBB3** i **MEMFC** bloka *brojači* generiše se aktivna vrednost signala **ClearD** bloka *indikatori*, pa se pri pojavi signala takta indikator D bloka *indikatori* adresiranog seta i to ulaza nula ili jedan keš memorije ***KEŠ*** postavlja na neaktivnu vrednost. U slučaju operacije upisa indikator D adresiranog seta i to ulaza nula ili jedan keš memorije ***KEŠ*** se postavlja na neaktivnu vrednost tek u koraku $step_7$ po dovlačenju novog bloka iz memorije ***MEM*** u ulaz nula ili jedan adresiranog seta keš memorije ***KEŠ***.

Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku $step_4$.

Pri aktivnoj vrednosti signala **MEMFC** na signal takta se prelazi iz koraka $step_4$ u korak $step_3$, $step_1$ ili $step_2$. U korak $step_3$ se prelazi za operacije upisa, selektivnog vraćanja i kompletnog vraćanja ukoliko nisu preneta iz keš memorije ***KEŠ*** u memoriju ***MEM*** sva četiri bajta bloka, pa brojač CNTBB ima sadržaj različit od tri. U korake $step_1$ i $step_2$ se prelazi ukoliko su preneti iz keš memorije ***KEŠ*** u memoriju ***MEM*** svi bajtovi bloka, pa brojač CNTBB ima sadržaj tri. U korak $step_1$ se prelazi u slučaju operacija selektivnog vraćanja i kompletnog vraćanja. Pošto je u koraku $step_3$ indikator D ulaza nula ili jedan adresiranog seta keš memorije ***KEŠ*** postavljen na neaktivnu vrednost, u koraku $step_1$ generisanjem aktivne vrednosti signala **clearV** bloka *indikatori* blok datog ulaza adresiranog seta će biti proglašen za nevažeći. U korak $step_2$ se prelazi u slučaju operacije upisa. !

$step_4$: **writeMEM**, **incMEMACC**,
if **MEMFC** *then* (**incABB**, **incCNTBB**),
if ((**CSC** + **CCC**)·**CNTBB3**·**MEMFC**) *then* **clearD**,
br (*if* $\overline{\text{MEMFC}}$
then $step_4$
else (*if* $\overline{\text{CNTBB3}}$
then $step_3$
else (*if* (**CSC** + **CCC**) *then* $step_1$ *else* $step_2$))),

! U korak $step_5$ se dolazi iz koraka $step_1$, $step_8$ ili $step_4$. Iz koraka $step_1$ se dolazi kada se za operacije čitanja ili upisa otkrije nesaglasnost ali i utvrdi da sadržaj bloka odabranog za zamenu nije modifikovan, pa se odmah prelazi na dovlačenje bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ*** u koracima $step_5$, $step_6$ i $step_7$. Iz koraka $step_8$ se dolazi u slučaju operacije čitanja po završenom prenosu bloka podataka iz bloka keš memorije ***KEŠ*** odabranog za zamenu u bafer bloka koji čine memorijski modul DATABUF bloka *datamam* i registar

TAGBUF bloka *tagmem*. U korak $step_8$ se prethodno dolazi iz koraka $step_1$ kada se za operaciju čitanja otkrije nesaglasnost ali i utvrdi da je sadržaj bloka odabranog za zamenu modifikovan. Iz koraka $step_4$ se dolazi u slučaju operacije upisa po završenom prenosu bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM*** u koracima $step_3$ i $step_4$. U korak $step_3$ se prethodno dolazi iz koraka $step_1$ kada se za operaciju upisa otkrije nesaglasnost ali i utvrdi da je sadržaj bloka odabranog za zamenu modifikovan.

Dovlačenje jednog bajta podatka iz memorije ***MEM*** u keš memoriju ***KEŠ*** se realizuje u koracima $step_5$, $step_6$ i $step_7$. Prilikom dovlačenja bloka podataka četiri puta mora da se prođe kroz ova tri koraka, pošto je veličina bloka četiri bajta. Ti prolasci se u tekstu nazivaju nulti, prvi, drugi i treći. Usvojeno je da se u prolasku 0 dovlači bajt bloka podataka za koji je generisana adresa, a da se u prolascima 1, 2 i 3 dovlače preostala tri bajta bloka podataka. U slučaju operacije čitanja je takođe usvojeno je da se bajt bloka podataka dovučen u prolasku 0 odmah prosledi i procesoru ***CPU***, da se procesoru ***CPU*** omogući da produži sa radom, ali i da se procesoru ***CPU*** signalizira da keš memorija ***KEŠ*** nije više zauzeta tek kada se dovuku i preostala tri bajta bloka podataka. Za realizaciju dovlačenja bajtova bloka podataka koriste se brojači ABB i CNTBB bloka *brojači* oba po modulu četiri. Brojač ABB svojom vrednošću određuje za prolaze 0 do 3 adrese bajtova u bloku, počev od bajta bloka za koji je generisana adresa. Brojač CNTBB svojom vrednošću određuje prolaze 0 do 3, počev od prolaza 0. Signali **CNTBB0**, **CNTBB1** i **CNTBB3** bloka *brojači* su signali dekodivanih stanja 0, 1 i 3 brojača CNTBB i svojim aktivnim vrednostima određuju da se radi o prolascima 0, 1 i 3, respektivno. U koraku $step_5$ se bezuslovno generiše aktivna vrednost signala **ldMAR** bloka *meminterfejs*, čime se na signal takta u registar MAR upisuje adresa memorije ***MEM***. U prolasku 0 to je adresa bajta podatka za koju je procesor ***CPU*** u keš memoriji ***KEŠ*** startovao operaciju čitanja. U prolasku 0 se u koraku $step_6$ podatak čita iz memorije ***MEM***, a u koraku $step_7$ upisuje u memorijski modul DATA0 ili DATA1 bloka *datamem*. U koraku $step_7$ se očitani podatak u slučaju operacije čitanja upisuje i u registar CDRRD bloka *cpuinterfejs*. U koraku $step_7$ se još i inkrementiraju brojači ABB i CNTBB. Prilikom prolaska 1 kroz korak $step_5$ ponovo se generiše aktivna vrednost signala **ldMAR**, čime se na signal takta u registar MAR upisuje adresa memorije ***MEM*** sa koje treba očitati sledeći bajt bloka. Međutim, sada je signal **CNTBB1** aktivan, pa se u slučaju operaciji čitanja, kada je signal **CRD** aktivan, generiše aktivna vrednost signala **CRP** bloka *cpuinterfejs*. Ovim signalom se bajt podatka očitan iz memorije ***MEM*** u prolasku 0 prosleđuje iz registra CDRRD bloka *cpuinterfejs* u procesor ***CPU***. Pored toga aktivna vrednost signala **CRP** omogućava procesoru ***CPU*** da ranije nastavi sa radom iako preostala tri bajta bloka tek treba da se dovuku iz memorije ***MEM*** u keš memoriju ***KEŠ*** u prolascima 1, 2 i 3. Međutim, signal zauzetosti **BUSY** bloka *cpuinterfejs* ostaje aktivan sve dok se preostala tri bajta bloka ne dovuku. Aktivna vrednost signala **BUSY** sprečava procesor ***CPU*** kome je dozvoljeno da ranije nastavi sa radom, da eventualno startuje novu operaciju čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja u keš memoriji ***KEŠ***.

Na signal takta se uvek prelazi iz koraka $step_5$ u korak $step_6$. !

$step_5$: **ldMAR**,
if (**CRD**·**CNTBB1**) *then* **CRP**,
br $step_6$,

! U korak $step_6$ se dolazi samo iz koraka $step_5$. U koraku $step_6$ se bezuslovno generiše aktivna vrednost signala **incMEMACC** bloka *brojači* čime se obezbeđuje da se na signal takta inkrementira sadržaj brojača **MEMACC**, koji određuje vreme pristupa memoriji ***MEM***.

Pri neaktivnoj vrednosti signala **MEMFC** bloka *brojači* se ostaje u koraku $step_6$.

Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDRRD** bloka *meminterfejs* čime se obezbeđuje da se na signal takta u registar MDRRD upiše vrednost očitana iz memorije ***MEM*** sa adrese određene sadržajem registra MAR bloka *meminterfejs*. Pri aktivnoj vrednosti signala **MEMFC** se na signal takta prelazi iz koraka $step_6$ u korak $step_7$. !

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

! U korak $step_7$ se dolazi samo iz koraka $step_6$. U koraku $step_7$ se bezuslovno generišu aktivne vrednosti signala **mxADATA**, **mxDIDATA** i **writeDATA** bloka *datamem* i **incABB** i **incCNTBB** bloka *brojači*. Aktivnom vrednošću signala **mxADATA** se na adresne linije memorijskih modula DATA0 i DATA1 dovodi adresa, aktivnom vrednošću signala **mxDIDATA** se na ulazne linije podataka memorijskih modula DATA0 i DATA1 dovodi bajt podatka očitan iz memorije ***MEM*** i aktivnom vrednošću signala **writeDATA** se vrši upis u memorijski modul DATA0 ili DATA1. Aktivnim vrednostima signala **incABB** i **incCNTBB** se na signal takta inkrementiraju sadržaji brojača ABB i CNTBB, respektivno. Prilikom prolaska 0 kroz korak $step_7$ brojač CNTBB ima vrednost nula, signal **CNTBB0** bloka *brojači* je aktivan, pa se u slučaju operaciji čitanja, kada je signal **CRD** aktivan, generišu aktivne vrednosti signala **mxDATA** i **ldCDRRD**. Pri aktivnoj vrednosti signala

mxDATA se u bloku *datamem* selektuje bajt podatka očitan iz memorije ***MEM***. Ovaj podatak se vodi na ulazne linije registra CDRRD bloka *cpuinterfejs* u koji se na signala takta i upisuje pri aktivnoj vrednosti signala **ldCDRRD**. Prilikom prolaska 3 kroz korak $step_7$ brojač CNTBB ima vrednost tri, signal **CNTBB3** bloka *brojači* je aktivan, pa se generišu aktivne vrednosti signala **writeTAG** bloka *tagmem*, **writeV** bloka *indikatori* i **adjFIFO** bloka *fifokola*. Aktivnom vrednošću signala **writeTAG** se u memorijski modul TAG0 ili TAG1 bloka *tagmem* upisuje broj grupe bloka koji je dovučen iz memorije ***MEM*** u memorijski modul DATA0 ili DATA1 bloka *datamem*. Aktivnom vrednošću signala **writeV** se na signal takta odgovarajući flip-flop V postavlja na aktivnu vrednost. Aktivnom vrednošću signala **adjFIFO** se na signal takta FIFO brojač odgovarajućeg seta ažurira. Pri aktivnoj vrednosti signala **CNTBB3** uslovno se generišu još signali **clCRD**, **writeD** i **clCWR**. U slučaju operacije čitanja, kada je signal **CRD** aktivan, generiše se, ukoliko je indikator D neaktivan, aktivna vrednost signala **clCRD** bloka *cpuinterfejs*. Aktivnom vrednošću signala **clCRD** se upisuje neaktivna vrednost u flip-flop CRD i operacija čitanja završava. Pored toga i signal **BUSY** bloka *cpuinterfejs* se postavlja na neaktivnu vrednost, tako da će procesor ***CPU*** moći da startuje novu operaciju čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja. Ukoliko je signal D aktivan, flip-flop CRD se postavlja na neaktivnu vrednost i operacija čitanja završava tek pošto se u koracima $step_9$ i $step_{10}$ izvrši prebacivanje bloka podataka iz bafera bloka u memoriju ***MEM***. U slučaju operacije upisa, kada je signal **CWR** aktivan, generišu se aktivne vrednosti signala **writeD** bloka *indikatori* i **clCWR** bloka *cpuinterfejs*, pa se na signal takta upisuje aktivna vrednost u odgovarajući flip-flop D i neaktivna vrednost u flip-flop CWR. Postavljanjem flip-flopa CWR na neaktivnu vrednost i signal zauzetosti keš memorije **BUSY** postaje neaktivan. Time je operacija upisa kompletirana.

Pri neaktivnoj vrednosti signala **CNTBB3** se na signal takta prelazi iz koraka $step_7$ na korak $step_5$. Pri aktivnoj vrednosti signala **CNTBB3** se na signal takta prelazi iz koraka $step_7$ na korak $step_9$ ili na korak $step_0$. Na korak $step_9$ se prelazi samo u slučaju operacije čitanja kada je sadržaj bloka podataka koji je odabran za zamenu modifikovan pa njegov sadržaj treba u koracima $step_9$ i $step_{10}$ prebaciti iz bafera bloka u memoriju ***MEM***. Na korak $step_0$ se prelazi u svim drugim situacijama. !

$step_7$: **mxADATA**, **mxDIDATA**, **writeDATA**, **incABB**, **incCNTBB**,
if (**CRD** · **CNTBB0**) *then* (**mxDATA, ldCDRRD**),
if **CNTBB3** *then* (**writeTAG**, **writeV**, **adjFIFO**),
if (**CRD** · $\overline{\mathbf{D}}$ · **CNTBB3**) *then* **clCRD**,
if (**CWR** · **CNTBB3**) *then* (**writeD**, **clCWR**),
br (*if* $\overline{\text{CNTBB3}}$ *then* $step_5$ *else* (*if* **CRD**·**D** *then* $step_9$ *else* $step_0$)),

! U korak $step_8$ se dolazi samo iz koraka $step_1$ i to samo za operaciju čitanja u slučaju da je utvrđeno da nema saglasnosti i da je sadržaj bloka keš memorije ***KEŠ*** odabranog za zamenu modifikovan.

U koraku $step_8$ se vrši prebacivanje bajtova bloka iz memorijskog modula DATA0 ili DATA1 u memorijski modul DATABUF bloka *datamem* i broja grupe iz memorijskog modula TAG0 ili TAG1 u registar TAGBUF bloka *tagmem*. Memorijski modul DATABUF i registar TAGBUF čine bafer bloka. Prebacivanje četiri bajta bloka se realizuje u četiri periode signala takta. U koraku $step_8$ se bezuslovno generišu aktivne vrednosti signala **mxADATA** i **writeDATABUF** bloka *datamem* i **incABB** i **incCNTBB** bloka *brojači*. Aktivnom vrednošću signala **mxADATA** se kroz odgovarajući multiplekser na adresne linije memorijskih modula DATA0 i DATA1 propušta sadržaj brojača ABB. Aktivnom vrednošću signala **writeDATABUF** se bajt podatka očitan iz memorijskog modula DATA0 ili DATA1 upisuje u memorijski modul DATABUF. Aktivnim vrednostima signala **incABB** i **incCNTBB** se pri pojavi signala takta inkrementiraju brojači ABB i CNTBB. Prebacivanje broja grupe se realizuje u četvrtoj periodi signala takta, kada je aktivna vrednost signala **CNTBB3**. Tada se generiše aktivna vrednost signala **ldTAGBUF** kojom se broj grupe pročitan iz memorijskog modula TAG0 ili TAG1 na signal takta upisuje u registar TAGBUF.

Pri neaktivnoj vrednosti signala **CNTBB3** se ostaje u koraku $step_8$, dok se pri aktivnoj vrednosti prelazi u korak $step_5$, radi dovlačenja bajtova bloka iz memorije ***MEM*** u keš memoriju ***KEŠ***.

$step_8$: **mxADATA**, **writeDATABUF**, **incABB**, **incCNTBB**,
if **CNTBB3** *then* **ldTAGBUF**),
br (*if* $\overline{\text{CNTBB3}}$ *then* $step_8$ *else* $step_5$),

! U korak $step_9$ se dolazi samo iz koraka $step_7$ i to samo kada je u slučaju operacije čitanja otkriveno da nema saglasnosti i da je sadržaj bloka keš memorije ***KEŠ*** odabranog za zamenu modifikovan.

U koracima $step_9$ i $step_{10}$ se realizuje vraćanje bajtova bloka iz memorijskog modula DATABUF bloka *datamem* u memoriju ***MEM***. U koraku $step_9$ se bezuslovno generišu aktivne vrednosti signala **ldMDRWR**, **mxMAR** i **ldMAR** bloka *meminterfejs*. Aktivnom vrednošću signala **ldMDRWR** se obezbeđuje da se na signal takta podatak $DATARD_{7...0}$ očitan iz memorijskog modula DATABUF upiše u registar MDRWR. U bloku

meminterfejs se aktivnom vrednošću signala **mxMAR** kroz multiplekser propušta vrednost $TAGOUT_{11...0}$ očitana iz memorijskog modula TAGBAF bloka *tagmem*, dok se aktivnom vrednošću signala **ldMAR** na signal takta u registar MAR upisuje formirana adresa.

Na signal takta se bezuslovno prelazi iz koraka $step_9$ u korak $step_{10}$. !

$step_9$: **ldMDRWR**, **mxMAR**, **ldMAR**,
br $step_{10}$,

! U korak $step_{10}$ se dolazi samo iz koraka $step_9$.

U koraku $step_{10}$ se bezuslovno generišu aktivne vrednosti signala **writeMEM** bloka *meminterfejs* i **incMEMACC** bloka *brojači*. Aktivnom vrednošću signala **writeMEM** se sadržaj registra MDRWR bloka *meminterfejs* upisuje u memoriju ***MEM*** na adresi određenoj sadržajem registra MAR bloka *meminterfejs*. Aktivnom vrednošću signala **incMEMACC** se obezbeđuje da se pri pojavi signala takta inkrementira sadržaj brojača MEMACC bloka *brojači* koji određuje vreme pristupa memoriji. Pri aktivnoj vrednosti signala **MEMFC** bloka *brojači* generišu se aktivne vrednosti signala **incABB** i **incCNTBB** koje obezbeđuju da se pri pojavi signala takta inkrementiraju brojači ABB i BBB, respektivno. Kada su aktivni i signal **MEMFC** i signal **CNTBB3**, generiše se aktivna vrednosti signala **clearD** i **clCRD**. Pri aktivnoj vrednosti signala **clearD** se na signal takta flip-flop D odgovarajućeg ulaza adresiranog seta postavlja na neaktivnu vrednost. Aktivnom vrednošću signala **clCRD** se omogućuje da se na signal takta flip-flop CRD postavi na neaktivna vrednost. Time se obezbeđuje da se i signal **BUSY** postavi na neaktivnu vrednost, čime se procesoru ***CPU*** omogućuje da startuje novu operaciju čitanja, upisa, selektivnog vraćanja ili kompletnog vraćanja. Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku $step_{10}$. Pri aktivnoj vrednosti signala **MEMFC** iz koraka $step_{10}$ se prelazi u korak $step_9$ ukoliko je signal **CNTBB3** neaktivan, odnosno u korak $step_0$ ukoliko je signal **CNTBB3** aktivan. !

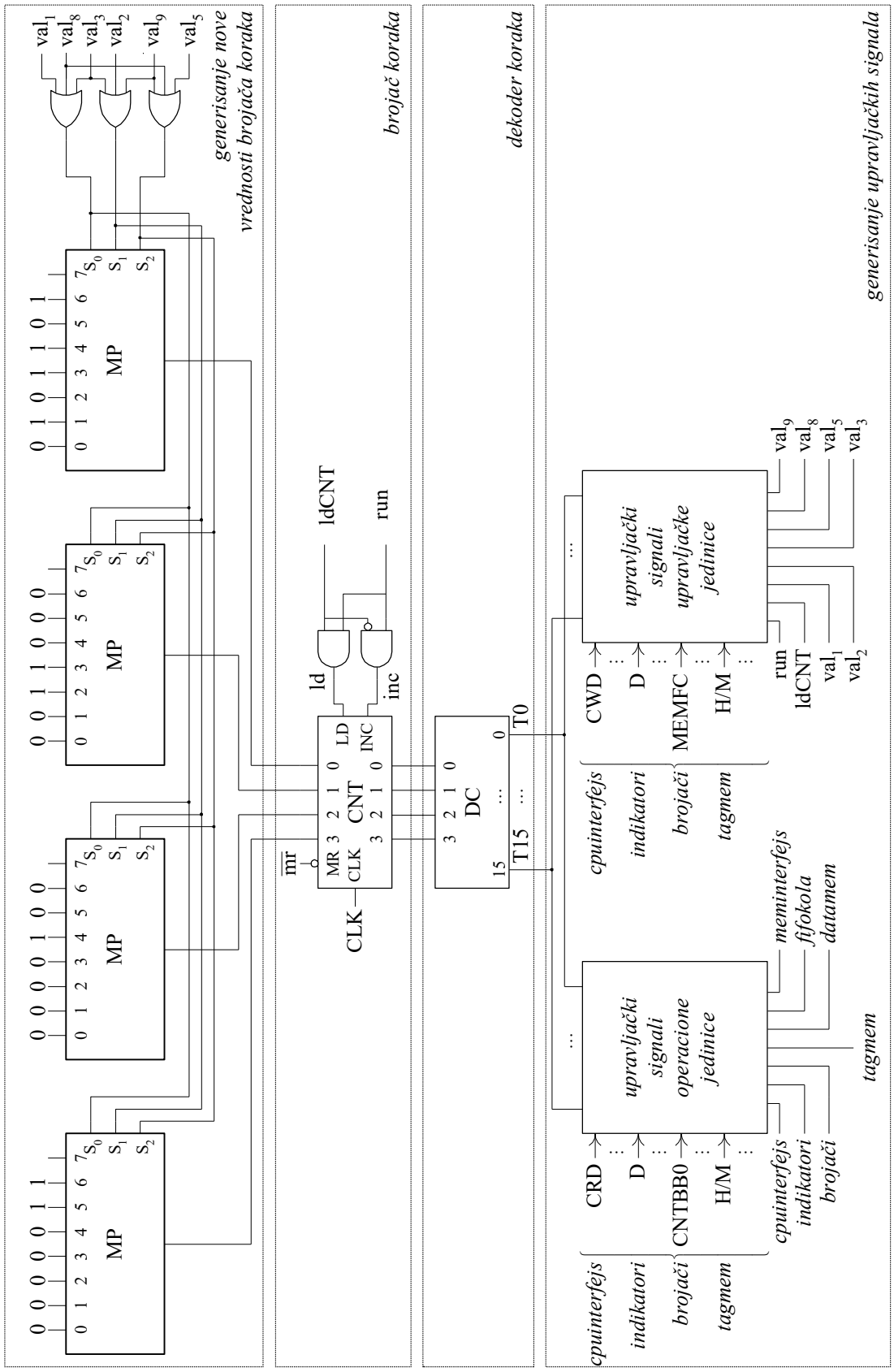
$step_{10}$: **writeMEM**, **incMEMACC**,
if **MEMFC** *then* (**incABB**, **incCNTBB**),
if (**CNTBB3** · **MEMFC**) *then* (**clearD**, **clCRD**),
br (*if* $\overline{\text{MEMFC}}$ *then* $step_{10}$ *else* (*if* $\overline{\text{CNTBB3}}$ *then* $step_9$ *else* $step_0$)),

1.2.3.3.2.3. Struktura upravljačke jedinice

Struktura upravljačke jedinice ožičene realizacije je prikazana na slici 82. Upravljačka jedinica 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.



Slika 82 Struktura upravljačke jedinice

1.2.3.3.2.3.1. Blok generisanje nove vrednosti brojača koraka

Blok *generisanje nove vrednosti brojača koraka* se sastoji od multipleksera MP i služi za generisanje i selekciju vrednosti koju treba upisati u brojač CNT. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog generisanja upravljačkih signala. Analizom sekvence upravljačkih signala po koracima (poglavlje 1.2.3.3.2.2) se utvrđuje da su 0, 1, 2, 3, 5, 8 i 9 vrednosti koje treba upisati u brojač CNT. Te vrednosti su ožičene na ulazima 0, 1, 2, 3, 4, 5 i 6 multipleksera, dok se ulaz 7 ne koristi. Selekcija jedne od tih sedam vrednosti se postiže odgovarajućim vrednostima signala $\mathbf{val_1}$, $\mathbf{val_2}$, $\mathbf{val_3}$, $\mathbf{val_5}$, $\mathbf{val_8}$ i $\mathbf{val_9}$. Ako su svi signali neaktivni kroz multipleksere se propušta vrednost 0, a aktivnom vrednošću samo jednog od ovih signala kroz multipleksere se propuštaju vrednosti 1, 2, 3, 5, 8 i 9, respektivno.

1.2.3.3.2.3.2. Blok brojač koraka

Blok *brojač koraka* se sastoji od brojača CNT. Brojač koraka 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 sekvence upravljačkih signala po koracima (poglavlje 1.2.3.3.2.2). 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 vrši se upis nove vrednosti u brojač CNT. 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 CNT 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 ne menja se vrednost brojača CNT. Ovaj režim rada se obezbeđuje neaktivnim vrednostima signala **inc** i **ld**. Da bi ovi signali bili neaktivni potrebno je da signal **run** bude neaktivan. Signal **run** je uvek aktivan, sem kada treba obezbediti režim mirovanja, što se dešava u koraku:

- $step_0$ kada se čeka pojava signala startovanja operacije čitanja **PRQRD,** upisa **PRQWR**, selektivnog vraćanja **PRQSC** ili kompletnog vraćanja **PRQCC** bloka *cpu interfejs*,
- $step_1$ u slučaju operacije kompletno vraćanje kada se utvrdi da blok keš memorije ***KEŠ*** adresiran flip-flopom CES i brojačem CNTSET nije modifikovan i da se flip-flopom CES i brojačem CNTSET nije došlo do bloka iz ulaza jedan seta 3 koji je zadnji blok keš memorije ***KEŠ***,
- $step_4$ u slučaju operacija upisa, selektivnog vraćanja i kompletnog vraćanja kada se, prilikom vraćanja bloka podataka iz keš memorije ***KEŠ*** u memoriju ***MEM***, čeka pojava signala **MEMFC** bloka *brojači* kao indikacija da je obavljen upis bajta u memoriju ***MEM***,
- $step_7$ u slučaju operacija čitanja i upisa kada se, prilikom dovlačenja bloka podataka iz memorije ***MEM*** u keš memoriju ***KEŠ***, čeka pojava signala **MEMFC** bloka *brojači* kao indikacija da je obavljeno čitanje bajta iz memorije ***MEM***,

- $step_8$ u slučaju operacije čitanja kada se, pri prebacivanju bloka podataka iz keš memorije ***KEŠ*** u bafer bloka, čeka pojava signala **CNTBB3** bloka *brojači* kao indikacija da je upis zadnjeg bajta u bafer bloka obavljen i
- $step_{10}$ u slučaju operacije čitanja kada se, prilikom prebacivanja bloka podataka iz bafera bloka u memoriju ***MEM***, čeka pojava signala **MEMFC** bloka *brojači* kao indikacija da je obavljen upis bajta u memoriju ***MEM***.

1.2.3.3.2.3.3. Blok dekoder koraka

Blok *dekoder koraka* se sastoji od dekodera DC. Na ulaze dekodera DC dovode se signali sa izlaza brojača CNT bloka *brojač koraka*. Dekodovana stanja brojača CNT pojavljuju se kao signali **T0** do **T15** na izlazima dekodera DC. Svakom koraku iz sekvence upravljačkih signala po koracima (poglavlje 1.2.3.3.2.2) dodeljen je jedan od ovih signala i to koraku $step_0$ signal **T0**, koraku $step_1$ signal **T1**, itd.

1.2.3.3.2.3.4. Blok generisanje upravljačkih signala

Blok *generisanje upravljačkih signala* sadrži kombinacione mreže koje pomoću signala **T0** do **T10** koji dolaze iz bloka *dekoder koraka*, signala logičkih uslova koji dolaze iz blokova operacione jedinice (poglavlje 1.2.3.1.1) i saglasno sekvenci upravljačkih signala (poglavlje 1.2.3.3.2.2) generišu aktivne vrednosti upravljačkih signala. Postupak projektovanja kombinacionih mreža je identičan sa postupkom koji je objašnjen u poglavlju 1.2.1.1.2.4.4.

Blok *generisanje upravljačkih signala* generiše dve grupe upravljačkih signala i to:
- upravljačke signale operacione jedinice i
- upravljačke signale upravljačke jedinice.

1.2.3.3.2.3.4.1. Upravljački signali operacione jedinice

Upravljački signali operacione jedinice ***oper_jed*** se daju posebno za svaki blok operacione jedinice i to blok *cpuinterfejs*, blok *indikatori*, blok *brojači*, blok *tagmem*, blok *datamem*, blok *fifokola* i blok *meminterfejs*.

Upravljački signali bloka *cpuinterfejs* se generišu na sledeći način:
- **ldCDRRD** = **T1** · **H/M** · **CRD** + **T7** · **CNTBB0** · **CRD**,
- **clCRD** = **T2** · **CRD** + **T7** · **CRD** · **CNTBB3** · $\overline{\mathbf{D}}$ + **T10** · **CNTBB3** · **MEMFC**,
- **clCWR** = **T2** · **CWR** + **T7** · **CWR** · **CNTBB3**,
- **clCSC** = **T2** · **CSC**,
- **clCCC** = **T2** · **CCC** i
- **CRP** = **T1** · (**CRD** + **CSC** + **CCC**) + **T2** · **CRD** + **T5** · **CRD** · **CNTBB1**.

Pri njihovom generisanju koriste se signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:
- **H/M** — blok *tagmem*,
- **CRD** — blok *cpuinterfejs*,
- **CWR** — blok *cpuinterfejs*,
- **CSC** — blok *cpuinterfejs*,
- **CCC** — blok *cpuinterfejs*,
- **CNTBB0** — blok *brojači*,
- **CNTBB1** — blok *brojači*,
- **CNTBB3** — blok *brojači*,
- **D** — blok *indikatori* i

- **MEMFC** — blok *brojači*.

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

- **writeV** = **T7** · **CNTBB3**,
- **clearV** = **T1** · **CSC** · **H/M** · **D** + **T1** · **CCC** · $\overline{\mathbf{D}}$,
- **writeD** = **T2** · **CWR** i
- **clearD** = **T4** · (**CSC** + **CCC**) · **CNTBB3** · **MEMFC** + **T10** · **CNTBB3** · **MEMFC**.

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

- **CNTBB3** — blok *brojači*,
- **CRD** — blok *cpuinterfejs*,
- **CWR** — blok *cpuinterfejs*,
- **CSC** — blok *cpuinterfejs*,
- **CCC** — blok *cpuinterfejs*,
- **H/M** — blok *tagmem*,
- **D** — blok *indikatori* i
- **MEMFC** — blok *brojači*.

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

- **incABB** = **T2** · **CWR** · $\overline{\mathbf{H/M}}$ + **T4** · **MEMFC** + **T7** + **T8** + **T10** · **MEMFC**,
- **incCNTBB** = **T2** · **CWR** · $\overline{\mathbf{H/M}}$ + **T4** · **MEMFC** + **T7** + **T8** + **T10** · **MEMFC**,
- **incMEMACC** = **T4** + **T6** + **T10**,
- **incCNTSET** = **T1** · **CCC** · $\overline{\mathbf{D}}$ i
- **ldABB** = **T1** · (**CRD** · $\overline{\mathbf{H/M}}$ + **CWR** · $\overline{\mathbf{H/M}}$ + **CSC** · $\overline{\mathbf{H/M}}$ · **D**).

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

- **CWR** — blok *cpuinterfejs*,
- **H/M** — blok *tagmem*,
- **MEMFC** — blok *brojači*,
- **CCC** — blok *cpuinterfejs*,
- **D** — blok *indikatori*,
- **CRD** — blok *cpuinterfejs* i
- **CSC** — blok *cpuinterfejs*.

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

- **writeTAG** = **T7** · **CNTBB3** i
- **ldTAGBUF** = **T8** · **CNTBB3**.

Pri njegovom generisanju koristi se signal logičkog uslova koji dolazi iz bloka operacione jedinice i to:

- **CNTBB3** — blok *brojači*.

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

- **mxDIDATA** = **T7**,
- **mxADATA** = **T2** · **CWR** · $\overline{\mathbf{H/M}}$ + **T7** + **T8**,
- **writeDATA** = **T2** · **CWR** + **T7**,
- **writeDATABUF** = **T8** i

- **mxDATA** = **T7** · **CRD** · **CNTBB0**.

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

- **CWR** — blok *cpuinterfejs*,
- **H/M** — blok *tagmem*,
- **CRD** — blok *cpuinterfejs* i
- **CNTBB0** — blok *brojači*.

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

- **adjFIFO** = **T7** · **CNTBB3** i
- **clFIFO** = **T2** · **CCC**.

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

- **CNTBB3** — blok *brojači* i
- **CCC** — blok *cpuinterfejs*.

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

- **mxMAR** = **T3** + **T9**,
- **ldMAR** = **T3** + **T5** + **T9**,
- **ldMDRWR** = **T3** + **T9**,
- **ldMDRRD** = **T6** · **MEMFC** i
- **writeMEM** = **T4** + **T10**.

Pri njihovom generisanju koristi se signal logičkog uslova koji dolazi iz bloka operacione jedinice i to:

- **MEMFC** — blok *brojači*.

1.2.3.3.2.3.4.2. Upravljački signali upravljačke jedinice

Upravljački signali upravljačke jedinice su:

- **ldCNT** — signal čijom se aktivnom vrednošću, pod uslovom da je signal **run** aktivan, upisuje 0, 1, 2, 3, 5, 8 ili 9 u brojač CNT, a neaktivnom vrednošću, pod uslovom da je signal **run** aktivan, inkrementira tekuća vrednost brojača CNT,
- **run** — signal čijom se aktivnom vrednošću omogućuje, u zavisnosti od vrednosti signala **ldCNT,** ili upis nove vrednosti ili inkrementiranje vrednosti brojača CNT, a neaktivnom vrednošću, bez obzira na vrednost signala **ldCNT**, onemogućava promena vrednosti brojača CNT i
- $\mathbf{val_1}$, $\mathbf{val_2}$, $\mathbf{val_3}$, $\mathbf{val_5}$, $\mathbf{val_8}$ i $\mathbf{val_9}$ — signali koji obezbeđuju generisanje vrednosti 0, 1, 2, 3, 5, 8 i 9 za upis u brojač CNT.

Ovi upravljački signali se generišu na sledeći način:

- $\mathbf{ldCNT} = \mathbf{T1} \cdot (\mathbf{CRD} \cdot \overline{\mathbf{H/M}} + \mathbf{CWR} \cdot \overline{\mathbf{H/M}} \cdot \mathbf{D} + \mathbf{CSC} \cdot \mathbf{H/M} \cdot \mathbf{D} + \mathbf{CCC} \cdot \mathbf{D}) + \mathbf{T2} + \mathbf{T4} \cdot \mathbf{MEMFC} + \mathbf{T7} + \mathbf{T8} \cdot \mathbf{CNTBB3} + \mathbf{T10} \cdot \mathbf{MEMFC}$,
- $\mathbf{run} = \mathbf{T0} \cdot (\mathbf{PRQRD} + \mathbf{PRQWR} + \mathbf{PRQSC} + \mathbf{PRQCC}) + \mathbf{T1} \cdot ((\mathbf{CRD} + \mathbf{CWR} + \mathbf{CSC}) + \mathbf{CCC}\ (\mathbf{D} + \overline{\mathbf{D}} \cdot \mathbf{CLEAN})) + \mathbf{T2} + \mathbf{T3} + \mathbf{T4} \cdot \mathbf{MEMFC} + \mathbf{T5} + \mathbf{T6} \cdot \mathbf{MEMFC} + \mathbf{T7} + \mathbf{T8} \cdot \mathbf{CNTBB3} + \mathbf{T9} + \mathbf{T10} \cdot \mathbf{MEMFC}$,
- $\mathbf{val_1} = \mathbf{T4} \cdot (\mathbf{CSC} + \mathbf{CCC}) \cdot \mathbf{CNTBB3} \cdot \mathbf{MEMFC}$,

- $\mathbf{val_2}$ = **T4** · **CWR** · **CNTBB3** · **MEMFC**,
- $\mathbf{val_3}$ = **T1** · (**CWR** · $\overline{\text{H/M}}$ · **D** + **CSC** · **H/M** · **D** + **CCC** · **D**) +
  T4 · $\overline{\text{CNTBB3}}$ · **MEMFC**,
- $\mathbf{val_5}$ = **T1** · **CRD** · $\overline{\text{H/M}}$ · $\overline{\text{D}}$ + **T2** · **CWR** · $\overline{\text{H/M}}$ + **T7** · $\overline{\text{CNTBB3}}$ +
  T8 · **CNTBB3**,
- $\mathbf{val_8}$ = **T1** · **CRD** · $\overline{\text{H/M}}$ · **D** i
- $\mathbf{val_9}$ = **T7** · **CRD** · **D** · **CNTBB3** + **T10** · $\overline{\text{CNTBB3}}$ · **MEMFC**.

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

- **CRD** — blok *cpuinterfejs*,
- **CWR** — blok *cpuinterfejs*,
- **CSC** — blok *cpuinterfejs*,
- **CCC** — blok *cpuinterfejs*,
- $\overline{\text{H/M}}$ — blok *tagmem*,
- **D** — blok *indikatori*,
- **MEMFC** — blok *brojači*,
- **CNTBB3** — blok *brojači*,
- **PRQRD** — blok *cpuinterfejs*,
- **PRQWR** — blok *cpuinterfejs*,
- **PRQSC** — blok *cpuinterfejs*,
- **PRQCC** — blok *cpuinterfejs*,
- **MEMFC** — blok *brojači* i
- **CLEAN** — blok *brojači*.