

SADRŽAJ

1 Uvod.....	3
1.1 Opšte napomene.....	3
1.1.1 Tehnike preslikavanja	4
1.1.1.1 Direktno preslikavanje	4
1.1.1.2 Asocijativno preslikavanje.....	6
1.1.1.3 Set-asocijativno preslikavanje	8
1.1.2 Dimenzija bloka.....	10
1.1.3 Algoritam zamene.....	11
1.1.4 Ažuriranje operativne memorije	11
1.1.5 Poboljšanja rada keš memorija	12
1.1.6 Keš memorija i U/I uređaji	13
1.1.7 Performanse keš memorije.....	15
1.1.7.1 Smanjenje <i>hit time</i> -a	15
1.1.7.2 Mala i jednostavna keš memorija	15
1.1.7.3 Smanjenje <i>miss rate</i>	15
1.1.7.4 Smanjenje <i>miss penalty</i>	16
1.1.8 Karakteristike razdvojenih keševa podataka.....	17
1.1.8.1 Keš koncepti.....	17
1.1.8.2 Razlog podeljenosti keš memorija	18
1.1.8.3 Klasifikacija	19
1.1.8.4 Dizajn keš memorije	20
1.1.9 Neke organizacije keš memorija.....	20
1.1.9.1 Victim keš	20
1.1.9.2 Dual Data Cache	21
1.1.9.3 Pseudo asocijativni keš	22
2 Opis sistema keš memorije	23
2.1 Keš memorija sa asocijativnim preslikavanjem.....	23
2.2.1 Procesor CPU	24
2.2.1.1 Operaciona jedinica	25
2.2.1.2 Upravljačka jedinica	27
2.2.2 Memorija MEM	28
2.2.3 Keš memorija KEŠ	29
2.2.3.1 Operaciona jedinica keš memorije.....	30
2.2.3.2 Upravljačka jedinica	34
3 Opis softverskog alata IGoVSoEDS.....	37
3.1 Biblioteke kola i hijerarhijsko stablo modula	37
3.2. Rad sa generatorom signala i memorijskim modulima	38
3.3. Promena širine konektorima (<i>Expand Connectors...</i>)	39
3.4 Invertori na konektorima kola (<i>Object/Connector Not Sign</i>)	40
3.5 Prikaz linija signala i njihovo kreiranje i modifikacija	41
3.6 Dodavanje modula u drugu elektronsku digitalnu strukturu.....	43
3.7 Rad sa parametrima simulacije i upravljanje simulatorom.....	44
4 Prikaz keš memorije sa asocijativnim preslikavanjem	46

Simulacija računarskog sistema sa asocijativnom keš memorijom

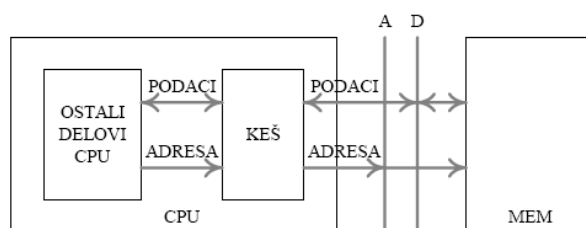
realizovane alatom IGoVSoDES	46
4.1 Procesor CPU	46
4.1.1 Blok keš interfejs	46
4.1.2 Blok generator operacija	47
4.1.3 Blok CPU upravljačka jedinica.....	48
4.2 Keš memorija KEŠ	48
4.2.1 CPU interfejs.....	48
4.2.2 Blok indikatori	49
4.2.3 Blok brojači.....	50
4.2.4 Blok tag memorija.....	50
4.2.5 Blok data memorija.....	51
4.2.6 Blok lru kola	52
4.2.7 Blok mem interfejs.....	52
4.2.8 Blok upravljačka jedinica	53
4.3 Memorija MEM	54
4.4 Prikaz procesa simulacije keš memorije sa asocijativnim preslikavanjem.....	55
5 Zaključak.....	73
6 Literatura.....	75

1 Uvod

U ovoj glavi se najpre daju neke opšte napomene o primenama i realizacijama keš memorija, a zatim se prikazuju korisnički zahtevi u pogledu realizacije keš memorije sa asocijativnim preslikavanjem.

1.1 Opšte napomene

Tehnološki razvoj procesora i memorija vremenom je doveo da velike razlike u brzinama rada ovih komponenti. S obzirom na to da ove komponente direktno komuniciraju, neophodno je bilo osmisлити dobro rešenje koje će smanjiti jaz u brzini i omogućiti brži rad računara. Sa porastom brzine rada procesora rasle su potrebe za pristupom memoriji, a vreme pristupa samoj memoriji usporavalo je procesor. Ideja kojom bi mogla da se smanje kašnjenja koja su nastajala usled nausaglašenosti broja pristiglih zahteva za korišćenje podataka i broja zahteva koji se mogu opslužiti u jedinici vremena, bila je da se postavi memorija koja bi bila posrednik između procesora i operativne memorije. Nazvana je **keš memorija**. Keš memorija je znatno manjeg kapaciteta, a vreme pristupa kešu je u velikoj meri manje u odnosu na vreme pristupa operativnoj memoriji, samim tim cena po bitu keš memorije je veća u odnosu na operativnu memoriju. Sa druge strane, ona čuva određeni broj podataka koji se nalaze i u memoriji i obraća se memoriji samo kada procesor zahteva podatke koji ona ne sadrži.



Kompletan rad realizovan je hardverski. Ova memorija čuva sadržaje operativne memorije kojima se procesor najčešće obraća. Prilikom izvršavanja instrukcija, kada procesor generiše adresu nekog podatka iz operativne memorije zbog čitanja podatka sa te lokacije ili upisa na tu lokaciju, prvo se vrši provera da li se podatak već nalazi u keš memoriji. Ako se podatak pronađe u keš memoriji, ne vrši se pristup operativnoj memoriji, nego se koristi podatak iz keša. U suprotnom, dovlači se podatak u keš memoriju i dalji rad se nastavlja samo sa keš memorijom. Na početku rada keš memorija je prazna, sa svakim novim zahtevom za podacima ona se puni, kada je keš memorija puna, velika je verovatnoća da će se u njoj već nalaziti podaci koje će biti potrebni procesoru u nekim narednim instrukcijama, a vreme pristupa biće smanjeno na minimum, jer sada procesor zapravo pristupa kešu.

Prilikom prebacivanja podatka veliku ulogu igra pretpostavka da će se u sukcesivnim instrukcijama generisati adrese podataka koji su često korišćeni i podataka koji se nalaze na sukcesivnim lokacijama u operativnoj memoriji. Ova pretpostavka se

Simulacija računarskog sistema sa asocijativnom keš memorijom

temelji na prostornom i vremenskom lokalitetu programa. *Prostorni lokalitet* podrazumeva da će se adrese koje će se uskoro generisati biti u blizini adrese koja je upravo generisana. Ovo je slučaj kog obrađivanja podataka koji su članovi nizova ili matrica. *Vremenski lokalitet* podrazumeva da će u određenom vremenskom intervalu biti generisane iste adrese koje i ne moraju da budu prostorno bliske. Ovo je slučaj kod generisanja adresa skalarnih promenljivih.

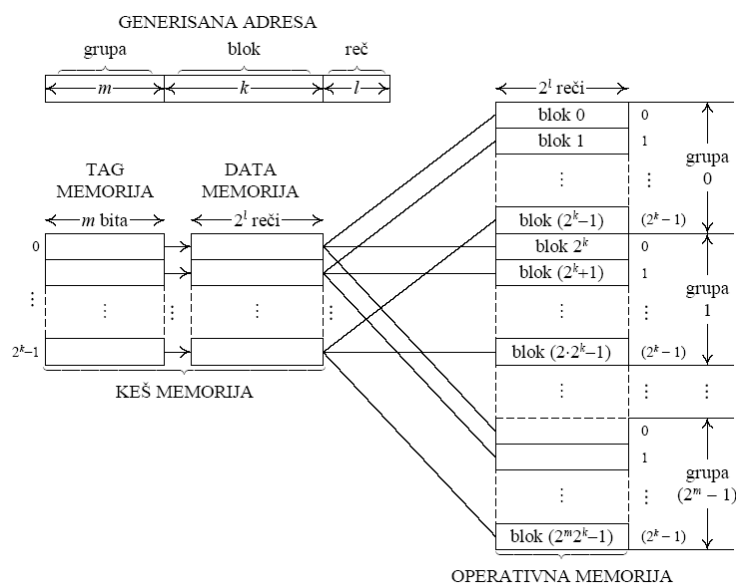
Da bi se kvalitet rada bio što bolji, neophodno je definisati određene karakteristike keš memorije. Ako se uzme u obzir prostorni lokalitet programa, potrebno je prebaciti u keš memoriju ne samo podatak koji se traži nego i ceo blok kome pripada, zbog toga se definiše **veličina bloka**. Ako želimo da prilagodimo keš memoriju vremenskom lokalitetu programa, potrebno je na pravi način voditi evidenciju o blokovima, uvesti odgovarajuću **tehniku preslikavanja** i usvojiti odgovarajući **algoritam zamene blokova**, prilikom generisanja adrese podatka koji se ne nalazi u punoj keš memoriji. **Evidencija o modifikovanim blokovima**, blokovima čije je sadržaj menjan, a nalaze se u keš memoriji, takođe je važna i neophodno je da postoji.

1.1.1 Tehnike preslikavanja

Tehnika preslikavanja predstavlja način vođenja evidencije o tome koji će se blokovi operativne memorije naći u kojim delovima keš memorije. Postoje sledeće tri tehnike preslikavanja:

- direktno preslikavanje
- asocijativno preslikavanje
- set-asocijativno preslikavanje

1.1.1.1 Direktno preslikavanje



Keš memorija sa direktnim preslikavanjem

Simulacija računarskog sistema sa asocijativnom keš memorijom

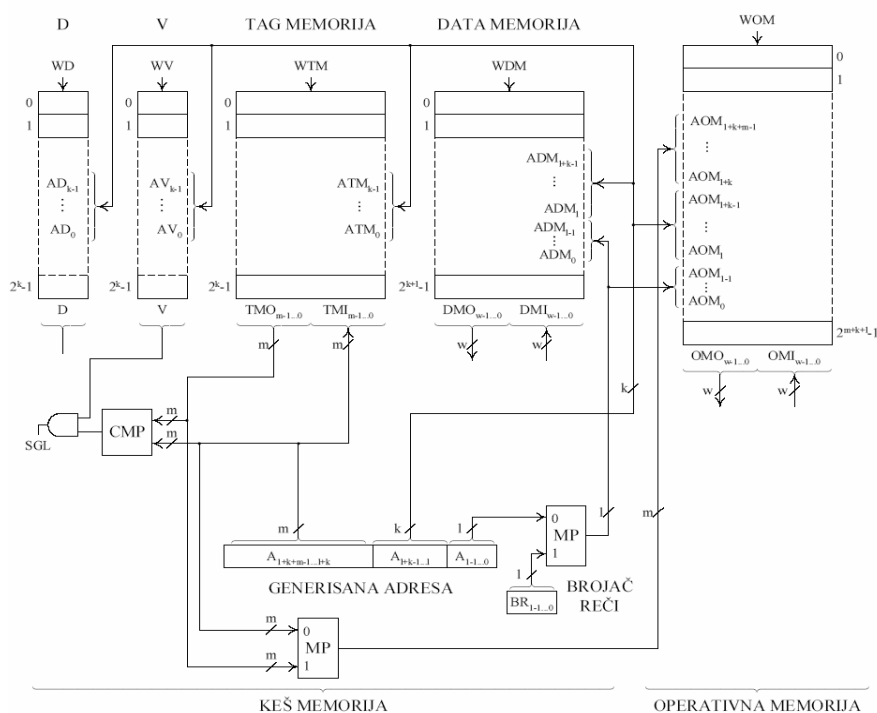
Kod ove tehnike direktnog preslikavanja keš memorija sastoji se iz:

- **DATA** memorije
- **TAG** memorije

U DATA memoriji čuvaju se sadržaji blokova koji su prebačeni iz operativne memorije u keš memoriju, dok se u TAG memoriji čuvaju brojevi grupa, koji se nazivaju *tag-ovi*, za blokove koji su preneti u keš memoriju iz operativne. Za realizaciju ovih memorija se koriste RAM memorijski moduli. Kaže se da keš memorija ima 2^k ulaza, ukoliko u data memoriju može da se smesti 2^k blokova operativne memorije. Za svaki ulaz data memorije postoji odgovarajući ulaz tag memorije.

Ukoliko posmatramo slučaj keš memorije sa 2^k ulaza realizovane u tehnici direktnog preslikavanja zamišlja se da je operativna memorije veličine 2^{m+k} blokova podeljena na 2^m grupa koje su sve veličine 2^k blokova, što znači da je veličina svake grupe jednaka veličini data memorije. Ukoliko se uzme da je dimenzija bloka 2^l reči tada generisana adresa ima sledeću strkturu:

- najnižih l bita određuje adresu reću u okviru bloka
- srednjih k bita određuje broj bloka unutar grupe
- najviših m bita određuje broj grupe u operativnoj memoriji kojoj blok pripada.



Detaljniji prikaz keš memorije sa direktnim preslikavanjem

Kod ove tehnike preslikavanja svaki blok ima tačno definisan ulaz keš memorije u koji se dovlači prilikom prebacivanja iz operativne memorije. Tako se u i -tom ulazu keš memorije može naći samo i -ti blok iz bilo koje od 2^m grupa. Pošto se u i -tom ulazu data memorije čuvaju samo sadržaji i -tih blokova u okviru grupa u i -tom ulazu tag memorije

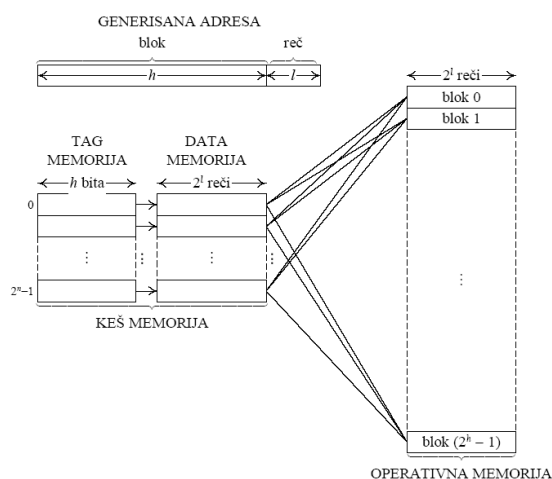
čuva se broj grupe kojoj blok, čiji se sadržaj čuva u istom ulazu data memorije, pripada. Kako postoji 2^m grupa, širina memorijske reči memorije tag je m bita.

1.1.1.2 Asocijativno preslikavanje

Kod tehnike asocijativnog preslikavanja keš memorija sastoji se iz sledećih delova:

- memorije **DATA** i
- memorije **TAG**.

U memoriju DATA smeštaju se blokovi preneti iz operativne memorije u keš memoriju, dok se u memoriji TAG čuvaju brojevi blokova koji se nalaze u DATA delu keš memorije, takozvani *tag-ovi*. Za realizaciju DATA memorije koriste se RAM memorija, dok se za realizaciju TAG memorije koristi asocijativna memorija. Veličina jednog ulaza u DATA memorii jednaka je veličini jednog bloka iz operativne memorije. Ako se u DATA memoriju može smestiti 2^n blokova iz operativne memorije, kaže se da keš memorija ima 2^n ulaza. Za svaki ulaz DATA memorije postoji odgovarajući ulaz TAG memorije.



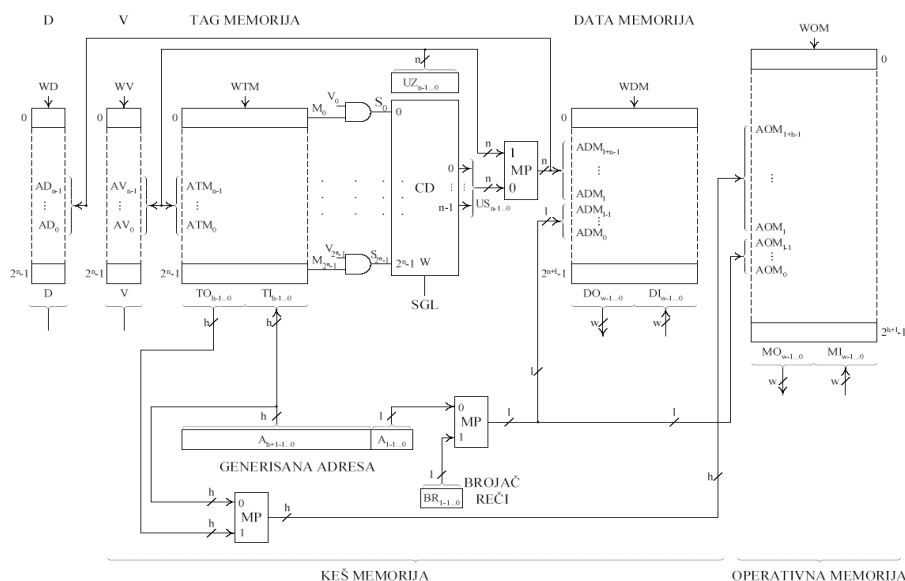
Keš memorija sa asocijativnim preslikavanjem

Kod keš memorije realizovane u tehnici asocijativnog preslikavanja pretpostavlja se da je operativna memorija podeljena na 2^h blokova, a da je svaki blok veličine 2^l reči, pa zbog toga generisana adresa ima sledeću strukturu:

- nižih l bita određuje adresu reči unutar bloka
- viših h bita određuje broj bloka u operativnoj memoriji.

U ovoj tehnici preslikavanja, svaki blok operativne memorije može se naći u bilo kom ulazu keš memorije prilikom dovlačenja. Uzimajući u obzir ovu činjenicu lako se dolazi do zaključka da se u i -tom ulazu keš memorije može naći sadržaj bilo kog bloka od 2^h blokova operativne memorije, a pošto se u memoriji TAG čuva broj bloka, širina memorijske reči memorije TAG treba da bude h bita.

Simulacija računarskog sistema sa asocijativnom keš memorijom



Detaljniji prikaz keš memorije sa asocijativnim preslikavanjem

Prilikom generisanja zahteva za čitanje od strane procesora, viših h bita generisane adrese vodi se na ulaz memorije TAG kako bi se utvrdilo da li sadržaj viših h bita generisane adrese poklapa sa bilo kojim sadržajem 2^n ulaza memorije TAG. Za svaki ulaz memorije TAG postoji poseban signal M_0 do M_{2^n-1} , takozvani signal poklapanja (Match), koji svojom aktivnom vrednošću ukazuje na činjenicu da li je došlo do podudaranja sadržaja nekog od ulaza memorije TAG sa generisanim bitima koji označavaju broj bloka u operativnoj memoriji, tj. da li je otkrivena saglasnost. Utvrđena saglasnost na nekom ulazu znači da se adresirana reč nalazi u bloku u ulazu u kome je otkrivena saglasnost, ali u DATA memoriji. Binarna vrednost broja ulaza je određena sa n bita sa izlaza koodera na osnovu signala M_0 do M_{2^n-1} . Signal saglasnosti **HIT** dobija se na izlazu **W** koodera samo u slučaju da neki od signala M_0 do M_{2^n-1} ima aktivnu vrednosti da je ulaz u kome je otkrivena saglasnost *validan*. Ukoliko postoji saglasnost, sa n bita izlaza koodera i l nižih bita generisane adrese adresira se reč u memoriji DATA i obavlja se čitanje. Ukoliko saglasnost ne postoji, na osnovu odabranog algoritma zamene blokova u keš memoriji, određuje se ulaz keš memorije za zamenu. Najpre se vrši prebacivanje sadržaja bloka koji je izabran za zamenu iz DATA memorije u operativnu memoriju, a broj bloka operativne memorije u koji se ovaj sadržaj upisuje određen je h bitima koje se nalaze u odgovarajućem ulazu memorije TAG koji je i izabran za zamenu. Zatim se iz operativne memorije u keš memoriju prebacuje ceo sadržaj bloka u kome se nalazi adresirani sadržaj generisanom adresom, sadržaj bloka smešta se u ulaz memorije DATA koji je odabran za zamenu, a u ulaz memorije TAG, takođe izabran za zamenu, upisuje se broj bloka koji je određen višim h bitima generisane adrese.

Pri generisanju zahteva za upis od strane procesora postojanje saglasnosti sa sadržajem keš memorije se ispituje na isti način kao kod operacije čitanja. Ukoliko postoji saglasnost, sa n bita sa izlaza koodera i nižih l bita generisane adrese, adresira se reč memorije DATA, a onda se obavlja i upis. Ako saglasnost ne postoji, postupak istovetan kao i kod operacije čitanja, najpre se bira ulaz keš memorije čije će sadržaj biti

Simulacija računarskog sistema sa asocijativnom keš memorijom

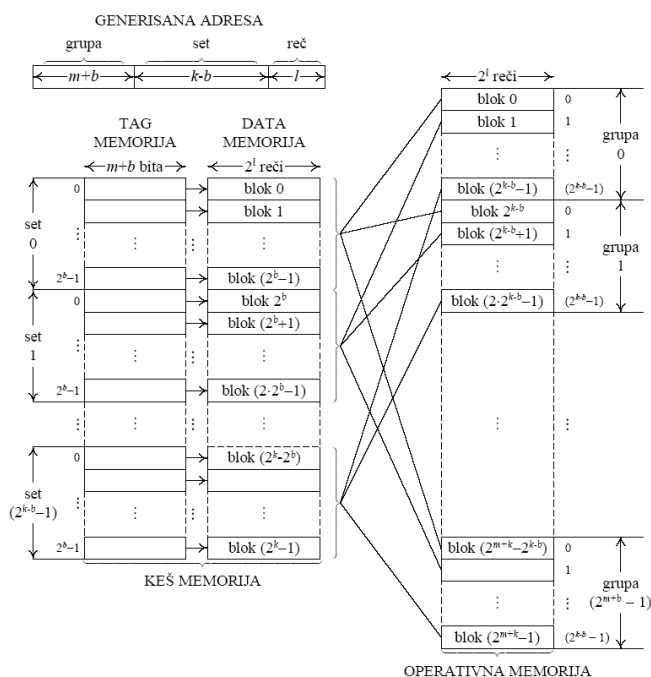
zamenjen, zatim se sadržaj tog ulaza DATA memorije prepisuje u blok operativne memorije čiji je broj određen sadržajem ulaza TAG memorije, odabranog za zamenu. Na kraju se u isti ulaz keš memorije dovlači ceo blok sa adresiranim sadržajem, reči se upisuju u memoriju DATA, a broj bloka u memoriju TAG, još jednom ispituje da li postoji saglasnost, utvrđuje da je došlo do saglasnosti sa sadržajem nekog od ulaza u memoriju TAG i realizuje se upis.

Do sada opisani mehanizam čitanja i upisa u keš memoriju realizovanom u tehnici asocijativnog preslikavanja je osnovni mehanizam rada sa keš memorijom, ali u praktičnim realizacijama postoji više varijanti ovog osnovnog mehanizma.

Prednost ove tehnike je pre svega u boljem popunjavanju keš memorije, ovde se bilo koji blok operativne memorije može smestiti u bilo koji ulaz keš memorije. Nedostak je u potrebi da se implementira neki od algoritama zamene i visoka cena asocijativne memorije koja se koristi za realizaciju memorije TAG, a koja zahteva složena kola za otkrivanje signala podudaranja.

1.1.1.3 Set-asocijativno preslikavanje

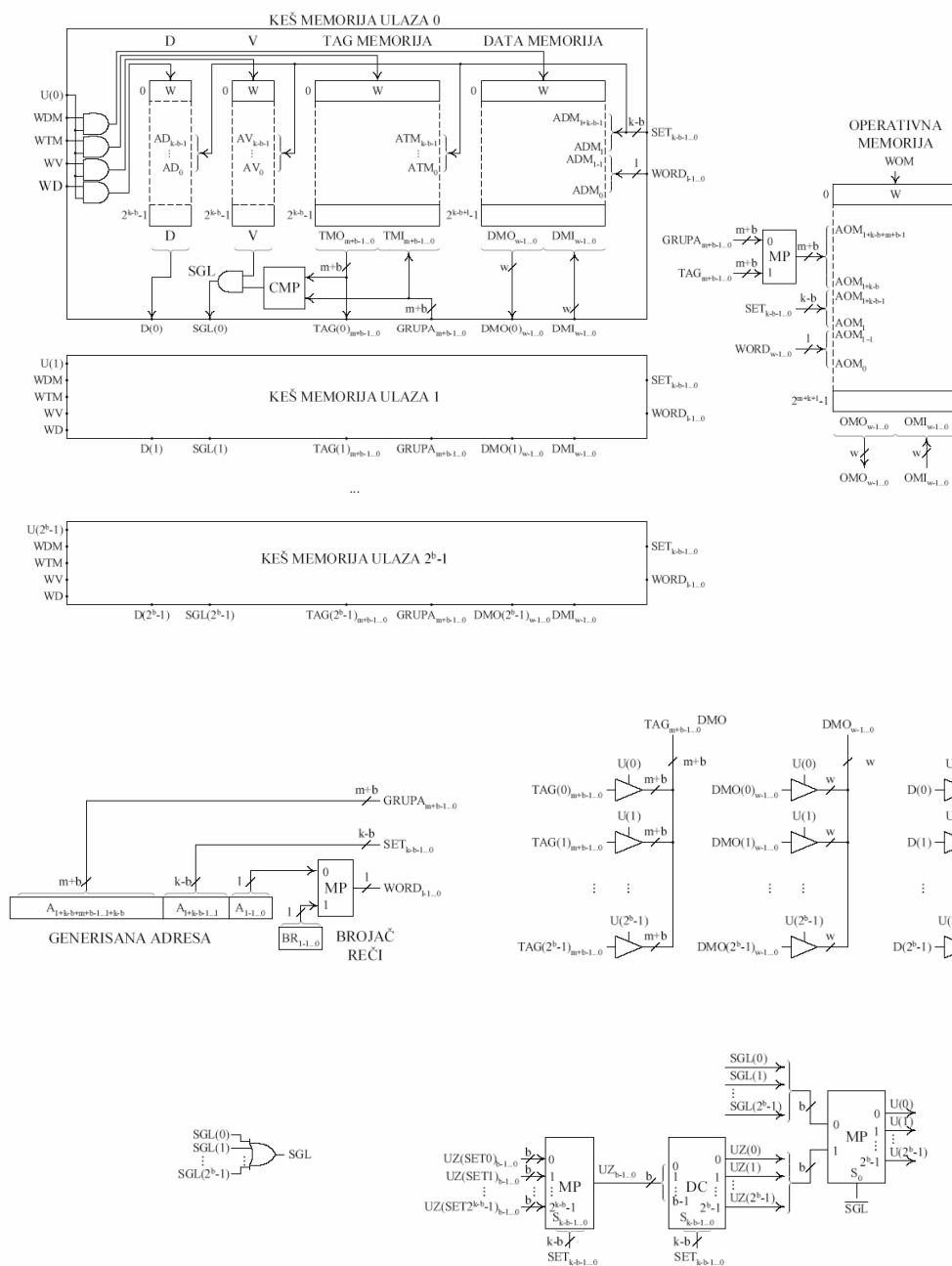
Tehnika set-asocijativnog preslikavanja je kombinacija prethodne dve tehnike. Kod ove tehnike preslikavanja uzima se da je keš memorija podeljena na dve ili više manjih identičnih keš memorija. Broj ovih manjih memorija je 2^b , a svaka od ovih manjih memorija realizovana je u tehnici direktnog preslikavanja. Operativna memorija je podeljena na grupe, a veličina svake grupe odgovara veličini jedne od manjih memorija. Broj bloka u grupi određuje ulaz u keš memoriji u koji se taj blok može prebaciti, kao



Keš memorija sa set-asocijativnim preslikavanjem

Simulacija računarskog sistema sa asocijativnom keš memorijom

kod tehnike direktnog preslikavanja. Pošto je uzeto da se keš memorija sastoji od 2^b manjih keš memorija, i -ti blok bilo koje grupe operativne memorije može da se preslika u i -ti blok bilo koje od 2^b manjih keš memorija. Time je stvoreno onoliko setova keš memorije koliko ima blokova u grupi operativne memorije, sa onoliko blokova po setu



Detaljniji prikaz keš memorije sa set-asocijativnim preslikavanjem

keš memorije koliko ima manjih keš memorija. Na nivou seta preslikavanje je direktno, jer je brojem bloka u grupi operativne memorije jednoznačno određen set keš memorije u

koji se dati blok preslikava. Na nivou seta preslikavanje je asocijativno, jer se dati blok operativne memorije može smestiti u bilo koji od 2^b blokova seta keš memorije.

Kod tehnike set-asocijativnog preslikavanja keš memorija podeljena je na 2^b manjih keš memorija i to *keš memorija 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 $(2b-1)$* , u kojoj se za sve setove čuvaju samo blokovi $(2b-1)$. Keš memorija sastoji se od 2^b **DATA** i **TAG** memorija. U DATA memoriju smeštaju se sadržaji blokova prenetih iz operativne u keš memoriju, a u TAG memoriju smeštaju se brojevi grupa kojima preneti blokovi pripadaju, a ovi brojevi nazivaju se *tag*-ovi. Za realizaciju DATA i TAG memorija, koristi se *RAM* memorija. Ukoliko u DATA memoriju 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 DATA i TAG memorija. DATA i TAG memorije 2^b manjih keš memorija obrazuju DATA memoriju i TAG memoriju 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} podeljena na 2^{m-b} grupa pri čemu veličina grupe odgovara veličini manje DATA memorije i iznosi 2^{k-b} blokova. Ukoliko se uzme da je veličina bloka 2^l reči tada generisana adresa ima strukturu:

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

1.1.2 Dimenzija bloka

Kod projektovanja keš memorije jako je važno na pravi način odabrati dimenziju bloka, tj. njegov kapacitet, jer ovaj podatak znatno utiče na performanse same keš memorije. Veći blokovi imaju određenih prednosti u pogledu veće prisutnosti podataka u kešu, kad se podaci dovuku u keš, procesor ukoliko treba da pristupa većem broju sekvencijalnih podataka, tj. podacima koji se nalaze na sekvencijalnim adresama, veća je verovatnoća da će biti u kešu i da će moći brže da im pristupi, nego kada bi se oni nalazili u više različitih blokova. Takođe postiže se ušteda u veličini TAG memorije, jer sada je širina ulaza u TAG memoriji manja, što je direktna posledica manjeg broja blokova u operativnoj memoriji, dok se u DATA memoriji takodje nalazi manje blokova, što olakšava realizaciju algoritma zamene blokova. Na osnovu do sada napomenutog, možemo da zaključimo da su blokovi većeg kapaciteta poželjni kod programa koji se sekvencijalno izvršavaju ili samo pristupaju sekvencijalnim adresama, jer se postižu bolje performanse procesora ukoliko se veći deo strukture ili podataka odjednom dovuče u keš.

Manji blokovi takođe imaju svojih prednosti. Prevedeno, vreme potrebno da se oni smeste u keš memoriju je kraće u odnosu na vreme potrebno da se dovuče veći blok. Zatim, kod manjih blokova manja je verovatnoća da će sadržati nepotrebne podatke, što kod blokova većih dimenzija može da predstavlja problem. Ukoliko je računar

projektovan tako da se prvo čeka da se ceo blok prenese iz operativne memorije u keš i da se tek nakon toga pristupa podatku, manji blokovi bi bili bolje rešenje u odnosu na veće.

Možemo da zaključimo da prednosti jednog izbora su mane drugog i obrnuto.

1.1.3 Algoritam zamene

Ukoliko se dogodi da se pri generisanju zahteva za čitanje ili upis od strane procesora utvrdi da se blok sa adresiranom reči ne nalazi u keš memoriji, potrebno je jedan od blokova koji se u tom trenutku nalazi u keš memoriju, vratiti u operativnu memoriju i u ulaz u kom se nalazio upisati blok sa zahtevanom reči. Ulaz keš memorije iz koga ćemo prebaciti blok u operativnu memoriju, a zatim i upisati sadržaj koji je zahtevan, biramo nekim od algoritama zamene. Algoritmi zamene realizuju se hardverski u keš memoriji.

Kod keš memorije sa direktnim preslikavanjem algoritam zamene je trivijalan, jer se već za vreme generisanja zahteva zna koji ulaz je odabran za zamenu. Kod keš memorije sa asocijativnim i set-asocijativnim preslikavanjem realizuje se algoritam zamene i on se kod tehnike asocijativnog preslikavanja primenjuje na sve ulaze keš memorije, jer se od svih ulaza bira jedan čiji se sadržaj menja, dok se kod tehnike set-asocijativnog preslikavanja primenjuje na blokove u okviru jednog seta.

Kod izbora algoritma zamene treba voditi računa da on bude takav da verovatnoća da se zahteva da se vrati blok iz operativne memorije u keš memoriju nakon što je taj blok bio vraćen u operativnu memoriju, bude minimalna. Sa druge strane cena hardvera kojim će se realizovati algoritam zamene treba da bude što je moguće niža. Ova dva zahteva su oprečna, jer cena realizacije algoritma koji će manje puta da zahteva prepisivanje blokova iz jedne u drugu memoriju je viša i obrnuto.

Postoji više pristupa kod korišćenja algoritma zamene. Jedan je da se zameni onaj blok iz keša memorije koji je najranije unet u keš memoriju. Ovaj algoritam naziva se **FIFO algoritam (First In First Out)**. Drugi pristup je da se zameni onaj blok keš memorije kome se najmanje skoro pristupalo. Ovaj algoritam naziva se **LRU algoritam (Least Recently Used)**. Hardver za realizaciju LRU algoritma, znatno je složeniji u poređenju sa hardverom za realizaciju FIFO algoritma. Pored navedenih algoritama dosta često se primenjuju i algoritam kojim se slučajno bira blok za zamenu korišćenjem jednog od postojećih generatora slučajnih brojeva.

1.1.4 Ažuriranje operativne memorije

Kod svakog generisanog zahteva za upis, blokovi u keš memoriji postaju modifikovani. Zbog promena koje nastaju potrebno je u određenim trenucima izvršiti ažuriranje blokova u keš memoriji, a čije se kopije, ali izmenjene nalaze u keš memoriji.

Postoje dva pristupa kojima se to postiže da sadržaji blokova u operativnoj memoriji budu ažurirani:

- ***upiši skroz*** (eng. *write-through* ili *store-through*)
- ***vrati nazad*** (eng. *write-back* ili *copy-back*)

Kod pristupa *upiši skroz*, pri svakom zahtevu za upis, istovremeno se vrši i upis u keš i operativnu memoriju. Time se garantuje da je sadržaj operativne memorije stalno ažuriran. Kod ovog pristupa ne postoji potreba za vođenje evidencije o modifikovanim blokovima, jer su sadržaji keš memorije i operativne memorije konzistentni. Kod pristupa *vrati nazad* pri svakom zahtevu za upis, ukoliko je podataka u keš memoriji, upis se vrši samo u keš memoriju. Na taj način sadržaj bloka u keš memoriji koji je modifikovan i njegov sadržaj u operativnoj memoriji nisu isti. Zbog toga je potrebno voditi evidenciju izmenama sadržaja blokova. Kasnije, kada treba dovući neki novi blok iz operativne memorije na mesto modifikovanog bloka, neophodno je prvo blok iz keša upisati u operativnu memoriju, kako bi se postiglo da sadržaj operativne memorije bude konzistentan. Kod pristupa *upiši skroz*, neophodno je i prilikom oduzimanja procesora nekom procesu, proveriti koji su blokovi u keš memoriji modifikovani i vratiti sve modifikovane blokove iz keš memorije u operativnu memoriju. Zahtev koji se javlja pored zahteva za čitanje i upis kod ovog pristupa je i zahtev za čišćenje keš memorije (eng. *flush*).

Prednost pristupa *upiši skroz* je u tome što je operativna memorija uvek ažurna. Nedostatak je u činjenici da se prilikom svakog upisa, pristupa i operativnoj memoriji, i time se opterećuje magistrala podataka. Prednost pristupa *vrati nazad* je upravo u činjenici da zahteva manji saobraćaj na magistrali, dok je mana u neažuriranoj operativnoj memoriji, kao i veliko vreme odziva keš memorije u slučaju promašaja.

Situacija kada u kešu ne postoji saglasnost kod operacije čitanje, može se rešiti na dva načina i to:

- ***dovuci blok*** (*write allocated*)
- ***ne dovuci blok*** (*no write allocated*)

Kod pristupa *dovuci blok*, blok se dovlači iz operativne memorije u keš memoriju, čime se sada omogućava da se otkrije saglasnost. Kod pristupa *ne dovuci blok*, blok se ne dovlači iz operativne u keš memoriju, već se upis vrši samo u operativnu memoriju.

1.1.5 Poboljšanja rada keš memorija

Radi poboljšanja funkcionalnosti keš memorije, moguće je uvesti promene u cilju skraćivanja vremena čitanja ili upisa u keš memoriju. Moguća poboljšanja vezana su za operaciju upisa. Kod operacije upisa procesoru se dozvoljava da nastavi sa izvršavanjem tekuće instrukcije bez zaustavljanja i čekanja na odgovor keš memorije da je upis završen. Na taj način dobija se paralelni rad procesora i keš memorije koja nezavisno od rada procesora obavlja upis. Keš memorija upisuje podatak, a procesor će izvršavati instrukciju. Keš ne prihvata novi zahtev za upis sve dok tekući ne završi.

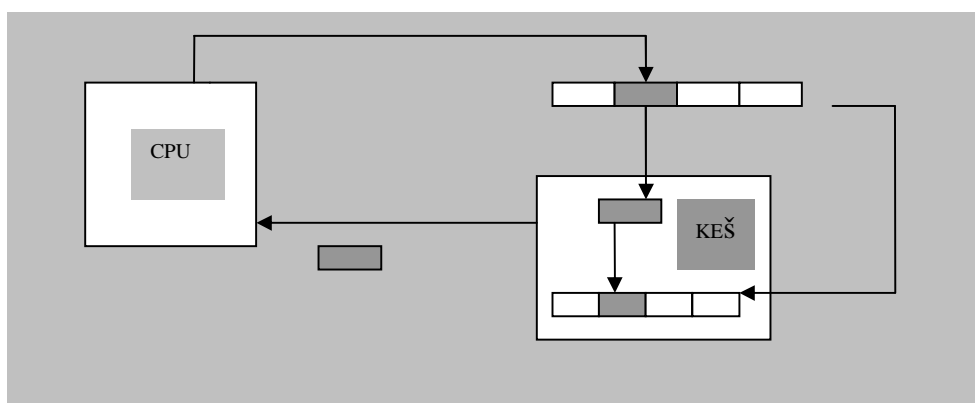
Simulacija računarskog sistema sa asocijativnom keš memorijom

Može se postići i ubrzanje operacije čitanja kada se traženi blok ne nalazi u keš memoriji. Ukoliko traženi blok nije u keš memoriji moguće je procesoru dostaviti traženi podatak, čim stigne iz operativne memorije, tako da on može ranije da nastavi sa izvršavanjem instrukcije, a paralelno se ostatak bloka prenosi u keš memoriju. I u ovom slučaju keš ne prihvata novu instrukciju čitanja sve dok ne završi sa tekućom. Ove tehnika naziva se *by-pass* (*premošćavanje*).

U slučaju vraćanja modifikovanog bloka, moguće je izvršiti ubrzavanje u smislu brzine rada. Da bi se proces ubrzao, moguće je modifikovani blok privremeno sačuvati u neki bafer, a potom po završetku obaveza prema procesoru, taj blok iz bafera upisati u operativnu memoriju. Kada se modifikovani blok smesti u bafer, traženi podatak se može smestiti u ulaz keš memorije, a da za to nije potrebno da se čeka da se modifikovani blok prvo nađe u operativnoj memoriji. Ovo poboljšanje se naziva *baferisanje*.

Sva navedena poboljšanja imaju za cilj da se procesor što manje zadržava prilikom obraćanja keš memoriji. Pri ovome se pretpostavlja da procesor neće neko vreme da se obraća keš memoriji, pa će do sledećeg zahteva keš memorija uspeti da obavi prethodno započetu operaciju do kraja.

Dakle, jedno od mogućih poboljšanja se ogleda u tome da se procesoru, prilikom operacije upisa dozvoli da on upiše u neki bafer, pa da produži sa izvršavanjem instrukcija. Time se paralelno obavlja upis u operativnu memoriju i tekuća instrukcija procesora. Naziv ovih tehnika je *baferovanje podataka* i *rani start procesora*. Slučajevi kada se procesoru odmah prosleđuje podatak, koji je zatražio, je *prosleđivanje* i *rani start procesora*. Ove tehnike se ogledaju u tome da se vrši dovlačenje bloka, od one reči koju je procesor i zatražio, njemu se ona prosledi, a potom se ostatak bloka prenese u keš memoriju (slika).



Princip dovlačenja prvo podatka koji je adresiran, a potom i ostatka bloka kome pripada

1.1.6 Keš memorija i U/I uređaji

Podaci se kod računara sa keš memorijom mogu naći u keš memoriji i u operativnoj memoriji. U sistemima gde je procesor jedini uređaj koji može da menja i čita podatke, a

Simulacija računarskog sistema sa asocijativnom keš memorijom

keš se nalazi između procesora i operativne memorije, nema opasnosti da procesor operiše sa starim (bajatom) kopijama podataka. Za razliku od ovog slučaja kod računara sa U/I uređajima i DMA kontrolerima, koji paralelno sa procesorom pristupaju memoriji, može se desiti da kopija nekog podataka u keš memoriji sa kojom radi procesor ne bude konzistentna sa kopijom u memoriji, sa kojom radi U/I uređaj. Ovaj problem se naziva problem *keš koherencije*.

Nastajanje problema keš koherencije zavisi od toga kako se ulaz/izlaz u računaru odvija. Moguće je da se odvija između U/I uređaja i keš memorije ili između U/I uređaja i memorije. Ukoliko se kod ulaza podaci stavljaju u keš, a kod izlaza uzimaju iz keša, tada U/I uređaji i procesor rade sa istom kopijom podatka. U ovom slučaju problem keš koherencije ne postoji. Međutim, ovde se U/I uređaji bore sa procesorom za pristup kešu, pa postoji opasnost da procesor čeka spori U/I uređaj da završi svoj ciklus. Pored toga može se desiti da U/I izbace pojedine blokove iz keš memorije koji su potrebni procesoru, pa ih on mora ponovo dovući u keš memoriju kako bi mogao da ih koristi. Ako se keš realizuje integrisan sa procesorom nastaje novi problem, a to je kako se vrši povezivanje U/I uređaja sa procesorom radi pristupa keš memoriji?

Sasvim je drugačija situacija u slučaju da se ulaz i izlaz obavlja preko memorije. Sada ne postoje gore navedeni problemi oko deljenja keš memorije. Jedino se postavlja pitanje oko problema vezanih za stare podatke. Ukoliko se koristi *write-through* tehnika, sadržaji keš i operativne memorija su ažurirani. Pošto su sadržaji identični, prilikom slanja sadržaja iz memorije u izlazni uređaj ne postoji opasnost od nekonzistentnih podataka. I pored nedostatka u opterećenju magistrale podataka, tehnika *write-through* veoma se često koristi.

Ukoliko se koristi *write-back* tehnika, situacija je daleko složenija i postoje dva rešenja, softversko i hardversko.

Softversko rešenje zahteva od operativnog sistema prenos podataka, iz keša u memoriju, svih modifikovanih blokova u opsegu adresa bafera memorije za izlaz, pre nego što se pošalju podaci iz memorije ka izlaznom uređaju. Tako se obezbeđuje da se u izlaznu jedinicu ne šalju bajati podaci. Naziv ove operacije je ispiranje keša (*flush*) i ona oduzima određeno vreme, čak i da podaci nisu u kešu, jer je potrebno proveriti, po svakom ulazu keša, da li je u opsegu adresa izlaza.

Hardversko rešenje hahteva posaban hardver keš memorije, koji na zahtev DMA za čitanje omogućuje da se provere sva „tag“ polja keša. Ukoliko se otkrije saglasnost i modifikacija bloka, hardver:

- generiše signal koji sprečava da memorija reaguje na signal čitanja
- šalje dati sadržaj iz keša na izlaznu jedinicu

Ovakvo rešenje može dovesti do usporavanja procesora, jer se sada pristupa „tag“ poljima i od strane procesora i dodatnog hardvera. Da bi se ovo izbeglo korisiti se udvajanje „tag“ polja keša. Jednima pristupa procesor, a drugima dodatni hardver.

Kod ulaza sa ulaznih jedinica problem se rešava ili softverski ili hardverski na isti način kao i kod “*write-back*” keša i izlaza. Softversko rešenje nalaže da se po kompletiranju ulaza od strane ulazne jedinice očisti keš. To takođe iziskuje dodatno vreme. Hardversko rešenje iziskuje udvajanje „tag“ polja, radi sprečavanja usporavanja keša, kao i proveru modifikacije blokova i „tag“ polja.

1.1.7 Performanse keš memorije

Mera performansi keš memorije je *miss rate* koja predstavlja vrednost procentualnih promašaja. Obično je manja od 5%. Kao moguća mera performanse keš memorije je i prosečno vreme pristupa memoriji (*average memory access time*), a izračunava se na sledeći način:

$$\text{average memory access time} = \text{hit time} + \text{miss rate} * \text{miss penalty}$$

hit time – vreme pristupa keš memoriji

miss rate – procenat promašaja

miss penalty – prosečno vreme pri promašaju

Dakle, prosečno vreme pristupa memoriji odražava brzinu hardvera, a po ovoj formuli zaključujemo da se performanse povećavaju smanjivanjem sva tri elementa. U narednim razmatranjima biće reči o tome kako realizovati poboljšanje performansi keš memorije.

1.1.7.1 Smanjenje *hit time*-a

Hit time predstavlja važan podatak za projektovanje procesora, jer ne utiče samo na prosečno vreme pristupa memoriji, već i periodu signala takta procesora. To je posledica višestrukog obraćanja procesora memoriji u toku izvršavanja jedne instrukcije, a ova obraćanja predstavljaju značajan deo ukupnog vremena izvršavanja instrukcije, tako da se perioda signala takta bira da predstavlja umnožak *hit time*-a. Smanjenje *hit time*-a moguće je ostvariti i korišćenjem male i jednostavne keš memorije.

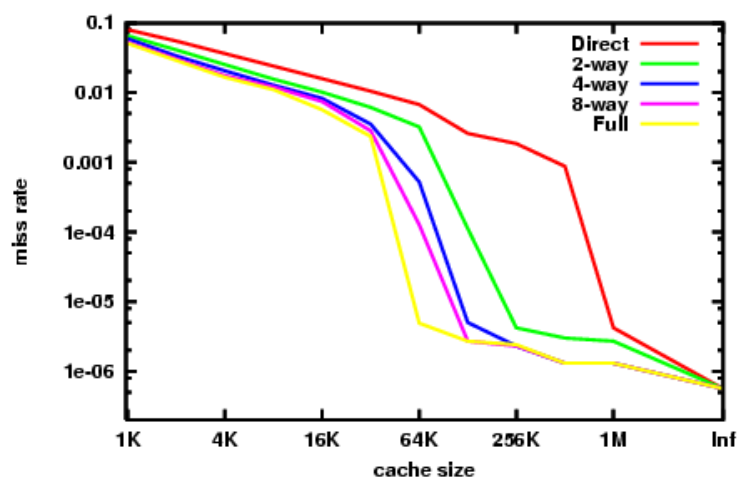
1.1.7.2 Mala i jednostavna keš memorija

Na *hit time* utiče vreme potrebno da se deo generisane adrese uzme i na osnovu njega izvrši provera tag polja u kešu, a zatim očita podatak. Najbrže se ostvaruje kod keša sa direktnim preslikavanjem, nešto sporije kod keša sa set-asocijativnim, a najsporije kod keša sa asocijativnim preslikavanjem.

1.1.7.3 Smanjenje *miss rate*

Miss-evi kod keš memorije mogu nastati iz tri razloga. *Obavezni miss*-evi - prvi pristup bloku uvek rezultira promašajem. Ovo se naziva *cold start miss* ili *first reference*

miss. *Miss*-evi zbog *kapaciteta* - keš memorija ima ograničen kapacitet i ne može da drži sve blokove koji su potrebni za vreme izvršavanja programa. *Promašaji zbog kapaciteta* nastaju zbog toga što se blokovi vraćaju iz keša u operativnu memoriju i posle ponovo dovlače iz operativne u keš memoriju. *Miss*-evi zbog *konflikata* - nastaju samo kod direktnog i set-asocijativnog preslikavanja. Ovi promašaji će se dešavati zato što neki blok može da bude vraćen iz keš u operativnu memoriju i kasnije ponovo dovučen u keš memoriju, ako se suviše mnogo blokova operativne memorije (koji pripadaju različitim grupama) preslikava na isti blok keš memorije (direktno preslikavanje) ili isti set keš memorije (set-asocijativno preslikavanje).



Miss rate kod različitih realizacija keš memorija različitih kapaciteta

Kako poboljšati *miss rate*:

- *Obavezni*—moglo bi da se poboljša povećanjem veličine bloka. Međutim veći blokovi povećavaju druge tipove *miss*-eva (na primer promašaje zbog kapaciteta i zbog konflikata).
- *Kapacitet*—jedino rešenje je ovde povećanje kapaciteta.
- *Konflikt*—sa povećanjem asocijativnosti smanjuju se ovi konflikti, međutim, rešenja sa povećanom asocijativnošću su hardverski skuplja, a povećava se i *hit time*.

Opšti zaključak je da mnoge tehnike koje smanjuju *miss rate* povećavaju *hit time* i/ili *miss penalty*. Zato se kod izbora ili primene neke tehnike mora voditi računa i o tome kako će se taj izbor reflektovati na druge faktore koji utiču na *prosečno vreme pristupa*, odnosno performanse keš memorije.

1.1.7.4 Smanjenje *miss penalty*

Smanjenje *miss penalty* direktno se postiže ako se uvedu dva bafera, za čitanje i upis. Pri pomašaju kod čitanje, ne prelazi se odmah na dovlačenje bloka iz operativne memorije, nego se prvo vrši provera da li se dati podatak nalazi u baferu za upis. Ukoliko se ispostavi da se nalazi, čeka se prvo da se podatak upiše kako bi se izbeglo dvostruko dovlačenje bloka sa podatkom. Ako se ne čeka upis za traženi podatak, on se odmah dovlači iz operativne memorije.

Simulacija računarskog sistema sa asocijativnom keš memorijom

Drugi način smanjivanja miss penalty je keš memorija sa podblokovima. Veliki blok čiji je kapacitet diktiran dozvoljenim kapacitetom **tag** dela podeli se u više podblokova, čija veličina odgovara optimalnoj veličini bloka. U keš memoriji **tag** se dodeljuje celom bloku, ali se pri promašaju dovlači samo potreban podblok. Ovo iziskuje da se za svaki blok mora voditi računa o *valid* bitima svih podblokova.

Dovlačenje tražene reči prvo, a zatim ostatka bloka i rani start procesora, takođe omogućavaju smanjenje *miss penalty*.

Kod pipeline-a često se koristi neblokirajući keš kog koga zahtev može da dođe iz drugog stepena, iako se *miss* iz prvog stepena još uvek nije kompletirao.

Možda jedna od danas jako rasprostranjenih tehnika je tehnika podele keš memorije u dva nivoa, tako da keš memorija prvog nivoa bude što je moguće manjeg kapaciteta i brža, a keš memorija drugog nivoa većeg kapaciteta i sporija, ali opet brža od operativne memorije. Ovako se postiže bolji kapacitet i veća brzina. Podaci se prvo traže u prvom nivou, zatim u drugom nivou i tek na kraju, ukoliko se ne pronađu u oba nivoa keš memorije, se dovlače iz memorije, ali je sada šansa da oni budu pronađeni u keš memoriji veća.

1.1.8 Karakteristike razdvojenih keševa podataka

U ovoj glavi razmatraju se neki od tipova keš memorija, koje su zasnovane na prostornom i vremenskom lokalitetu podataka. Razmotrićemo najpre keš koncepte, a zatim, pogledati neke specifične organizacije keš memorija. Razmotrićemo neke od tehnika, koje će predstavljati budućnost u razvoju keš memorija.

1.1.8.1 Keš koncepti

Namena keš memorije je da prikrije kašnjenje memorije, smanji zahtevnost u pogledu propusne moći glavne memorije, kao i da „udovolji“ vremenskom i prostornom lokalitetu programa.

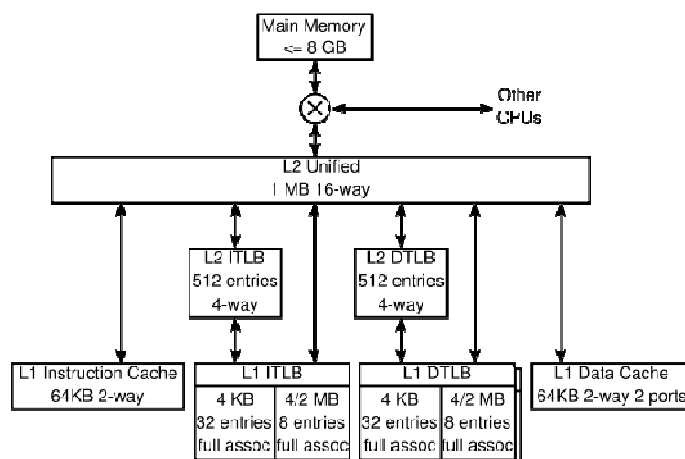
Lokaliteti – prostorni i vremenski.

Asocijativnost – direktno mapirane, set-asocijativne i potpuno asocijativne.

Miss tipovi – kapacitet, koherencija, konflikti i obavezni promašaji.

Keš je uobičajeno organizovana kao višestepena hijerarhija. Prvi stepen je podeljen na dve nezavisne organizacije, za instrukcije i podatke. Razlika je u ponašanju instrukcija, koje su uglavnom prostornog tipa, dok su podaci, i prostornog i vremenskog tipa. Ovde će biti razmatrane keš memorije podataka.

Simulacija računarskog sistema sa asocijativnom keš memorijom



Ilustracija obe specijalizacije slojevite strukture keš memorije AMD Athlon 64

1.1.8.2 Razlog podeljenosti keš memorija

Podaci ispoljavaju prostorni i vremenski lokalitet. Konvencionalna organizacija keša ne razlikuje različite tipove lokaliteta, već je neselektivni keš za sve podatke. Ova neselektivnost je uslov da se pojave:

- *nepotrebna pomeranja podataka kroz nivoe hijerherhije*
- *značano mešanje nepovezanih podataka u kešu*
- *unošenje nepotrebnih podataka u keš, što kao proizvod ima izbacivanje potencijalno korisnih podataka, kreirajući tako „zagađenje keša“*

Posledice koje se javljaju takvim ponašanjem su:

- *miss ratio*
- *memory access time*
- *memory bandwidth*

Ideja je da se podaci podele u više setova u okviru kojih će se ispoljavati ista svojstva i karakteristike podataka.

Na podelu su uticali sledeći faktori:

- *različiti tipovi lokaliteta i tipova podataka ispoljavaju različite zahteve u pogledu organizacije keša, tertiranja podataka...*
- *radi boljeg iskorišćenja vremenskog lokaliteta bolji su manji blokovi, dok za prostorni lokalitet su bolji veći*
- *za podatke sa prostornim lokalitetom bolje je pred-dohvatanje (prefetching), dok za podatke sa vremenskim lokalitetom pred-dohvatanje je beskorisno*
- *nemoguće je kreirati keš memoriju koja će objediniti ove osobine, prostornog i vremenskog lokaliteta, kao i za različite tipove podataka*

Zbog različitosti, a i povećanja performansi dizajnirani su višestruki keševi podataka sa različitim organizacijama. Svaki podset podataka stavljen je pojedinačni

podkeš koji je onakve realizacije koja na najbolji način podržava svojstva koja dati podaci ispoljavaju.

1.1.8.3 Klasifikacija

Zbog velike raznolikosti u idejama i rešenjima, veoma ih je teško razvrstati. Ukoliko dva dizajna pripadaju istoj grupi, to ne znači da su oni slični. Sakupljaju se informacije o uskladištenim keš blokovima, njihovih osobina i smeštaju u L1 (keš prvoga nivoa) keševe. Odluka u koji podkeš će se smestiti blok donosi se na osnovu ovih prikupljenih informacija. Takođe se mogu smestiti i u L2 (keš drugog nivoa) keš ili u pomoćnoj TRI (*table reuse information*) tabeli. Neki od dizajna keš memorije koriste informacije o blokovima, a neke i ih i ne koriste.

Keš memorije koje koriste informacije o blokovima (managing reuse information):

- Dual Data Caches
- STS
- NTS
- SSNS
- LSM
- Filter Data Caches

Keš memorije koje ne koriste informacije o blokovima (non-managing reuse information):

- Victim Cache
- Assist Cache
- Allocation by Conflict Scheme

Podkeševi mogu biti potpuno različite organizacije, počev od različitog kapaciteta bloka, pa do samih tehnika preslikavanja.

Različita asocijativnost sa istom veličinom bloka:

- Victim Cache
- Assist Cache
- NTS
- minimax

Ista asocijativnost sa različitom veličinom bloka:

- Dual Data Caches
- STS
- SS/NS
- Array/Scalar

1.1.8.4 Dizajn keš memorije

Odluke o dizajnu keš memorije koje se moraju doneti pre kreiranja su:

- *karakteristike podataka koje će biti smeštene u podkeševe*
- *organizacija keša*
- *putanja toka podataka između podkeševa*

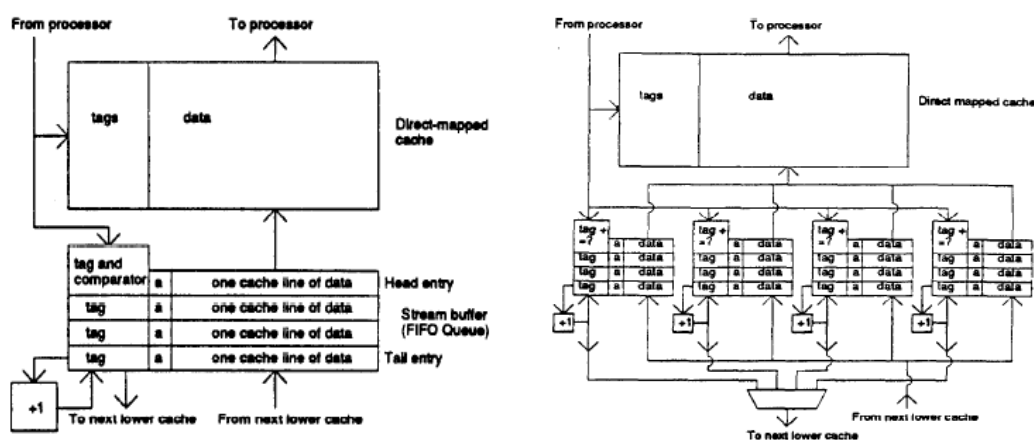
Proces dizajna predstavljen je u četiri koraka:

1. specifikacija tipova podataka, razdvajanje tipova podataka u setove, svaki set sa specifičnom organizacijom
2. odluka kada klasifikovati podatke, i kako će šema raditi
3. podešavanja karaktersitika keša, veličina keša, veličina bloka, algoritmi zamene
4. provera ideje, analitički model, pokretanje simulacija

1.1.9 Neke organizacije keš memorija

1.1.9.1 Victim keš

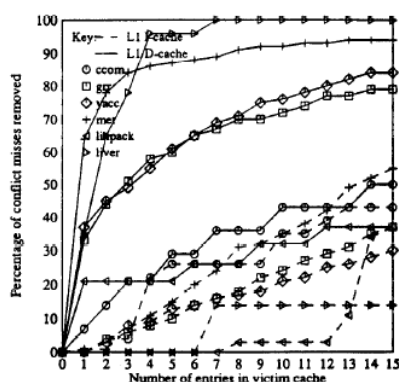
Performanse direktno mapiranih keš memorija su bolje nego set-asocijativne keš memorije, bez obzira na veliki broj konfliktnih promašaja. Ideja za dizajniranjem **Victim keša** potiče od nastojanja da se smanji broj promašaja i samim tim povećaju performanse. Pojava konfliktnih promašaja uslovljena je smanjenom asocijativnošću keša. Zato se izvršilo dodavanje male potpuno asocijativne keš memoriju, pored glavne direktno mapirane keš memorije, kako bi se povećala asocijativnost. Cilj asocijativnog keša je u tome da se zaustavi pojava konfliktnih linija podataka. Ovaj keš nazvan je **Victim keš** i sadrži blokove koji su bili „žrtve“ pri *miss*-evima. Pri *miss*-u se najpre vrši provera u **victim kešu**. Ako jeste, onda *keš blok* i *victim blok* menjaju mesto. Ako nije u **victim kešu** tek onda se ide u operativnu memoriju.



Šema Victim keša

Simulacija računarskog sistema sa asocijativnom keš memorijom

Dakle, pogodak u glavnoj keš memoriji znači da se pristup nastavlja normalno. Promašaj u glavnom kešu, pogodak u manjem kešu, blokovi glavnog i pomoćnog keša (victim keša) se zamenjuju. Promašaj u oba keša, referentni blok dolazi od L2 keša u glavni keš, konfliktni blok iz glavnog keša ide u pomoćni keš. Sličnu organizaciju poseduje i *Assist Cache*, jedina razlika je u konekcijama blokova podataka između podkeševa.

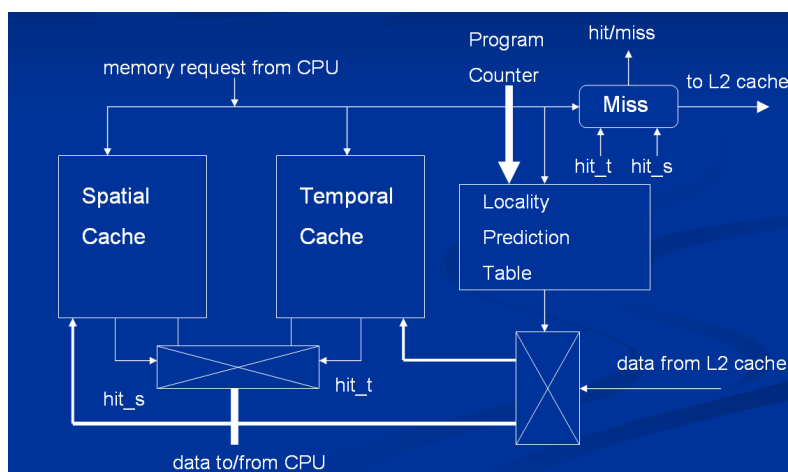


Procenat uklonjenih konfliktnih promašaja

1.1.9.2 Dual Data Cache

Dual Data keš koristi se za podatke koji ispoljavaju različite lokalitete, a nalaze se u jednoj keš memoriji koja može da bude pogodna za jednu grupu podataka, a za drugu grupu da smanjuje performanse i izaziva „zagađenje keša“.

Kod **Dual Data keš-a** podaci su podeljeni u dve nezavisne keš memorije. Jedna keš memorija namenjena je podacima koji ispoljavaju prostorni lokalitet, a druga podacima koji ispoljavaju vremenski lokalitet. Podatke sada usmeravamo na osnovi **LPT** (*Locality Prediction Table*) tabele.



Dual Data keš

Simulacija računarskog sistema sa asocijativnom keš memorijom

To je tabela koja „pamti“ informacije o najsvežijim *load/store* instrukcijama. Predikcija gde će se podatak smestiti može biti:

- *Prostorni*
- *Vremenski*
- *Bypass*
- *Definiciona vrednost*

Veličina linije prostornog keša, je nekoliko desetina bita i više, dok je za vremenski keš samo nekoliko bita. Dakle, prostorni lokalitet je iskorišćen ukoliko je linija keša što veća, dok je vremenski lokalitet iskorišćen pri manjoj veličini keš linije. Ukoliko se podaci nalaze u jednoj od podkeševa, podaci se čitaju i upisuju u taj podkeš. Međutim ukoliko se podatak nalazi u obe keš memorije, on se čita iz vremenskog keša, zbog manje keš linije, tj manjeg kapaciteta bloka (*Hit time je manji!*). Upis se međutim vrši u obe keš memorije. Ukoliko se podatak ne nalazi ni u jednoj od ovih keš memorija, podaci se dovlače iz keševa viših u hijerehiji, L2 keševa, i usmerava na osnovu predikcije.

1.1.9.3 Pseudo asocijativni keš

Pseudo asocijativni keš je pokušaj dobijanja *miss rate*-a set-asocijativnih i *hit time*-a direktnih memorija. Koristi se keš memorija sa direktnim preslikavanjem. Provera da li ima *hit* i ako ima pristup kao kod uobičajenog direktnog preslikavanja, ako nema još jedan ulaz u keš memoriji se proverava da li ima saglasnost. Pseudo asocijativna keš memorija ima sada **brzi** i **spori hit**. Opasnost je da mnogi **fast hit times** ne postanu **slow hit times** u pseudo asocijativnoj keš memoriji.

2 Opis sistema keš memorije

2.1 Keš memorija sa asocijativnim preslikavanjem

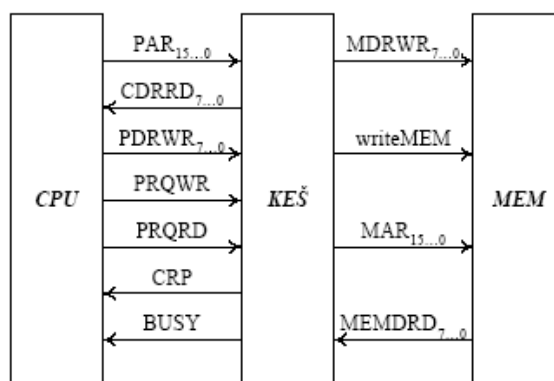
U ovom slučaju realizovana je keš memorija sa asocijativnim preslikavanjem. Dimenzija bloka na nivou koga se vrši preslikavanje je 4B. S obzirom da se radi o potpunoj asocijativnosti (**fully associative**), blokovi se prilikom dovlačenja iz operativne memorije u keš memoriju mogu smestiti u bilo koji ulaz keš memorije, neophodno je koristiti neki od algoritama zamene, u slučaju da je keš memorija puna, a adresirani podatak se ne nalazi u njoj. Ovde je realizovan algoritam LRU (Least Recently Used) koji od svih blokova keš memorije za zamenu bira onaj kome se najmanje skoro pristupalo, tj. onaj kome se najduže nije pristupalo. 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 upisa podataka on direktno upisuje u operativnu memoriju, ali ako postoji saglasnot upisuje se i u keš memoriju. Zbog toga se blok dovlači u keš memoriju samo u slučaju operacije čitanja ukoliko nije otkrivena saglasnost, dok kod operacije upisa, čak iako nema saglasnosti blok se ne dovlači u keš memoriju već se upis vrši u operativnu memoriju, kao što je već napomenuto.

Keš memorija ima 8 ulaza u koje se mogu smestiti blokovi, što na nivou cele keš memorije znači da je kapacitet dela gde se smeštaju podaci 32B, adresibilna jedinica je jedan bajt. Operativna memorija je kapaciteta 64 KB, pa je širina adrese 16 bita ($2^{16} = 2^{14}2^2 = 64KB$). Stoga operativnu memoriju možemo da posmatramo kao da je organizovana u 2^{14} blokova kapaciteta 2^2 bajta. Adresa operativne memorije može se podeliti na sledeći način:

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

Keš memorija je deo sistema koji se sastoji iz:

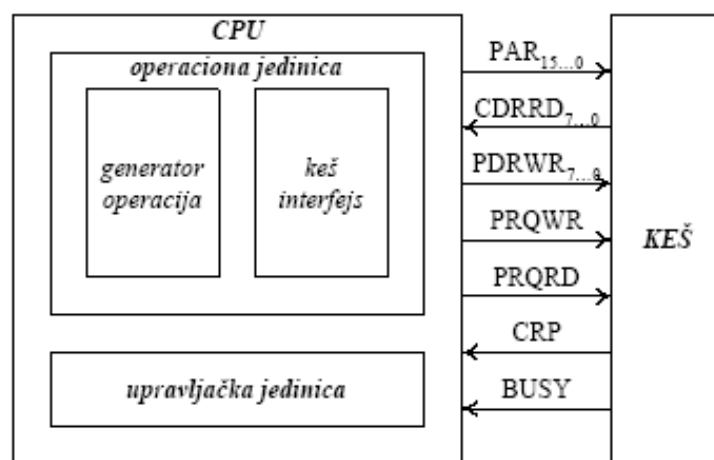
- procesora **CPU**
- memorije **MEM**
- keš memorije **KEŠ**.



2.2.1 Procesor CPU

Procesor **CPU** se obraća keš memoriji **KEŠ** kada treba da čita ili upiše podatak u keš memoriju **KEŠ**, tj. kad treba da izvrši operaciju čitanja ili upisa. Kada se generiše adresa memorije **MEM** treba izvršiti jednu od ove dve operacije. Ovom prilikom razmenjuju se određeni signali između procesora **CPU** i keš memorije **KEŠ**, ovi signali prikazani su na slici.

Kod operacije čitanja ili operacije upisa, procesor **CPU** po linijama $PAR_{15..0}$ šalje 16-bitnu adresu keš memoriji **KEŠ**.



U slučaju da se radi o operaciji čitanja odmah se generiše aktivna vrednost signala **PRQRD** koja traje jednu periodu signala takta. Keš memorija **KEŠ** zatim vraća očitani 8-bitni podatak po linijama $CDRRD_{7..0}$. Keš memorija **KEŠ** šalje i signale **CRP** i **BUSY**. Aktivnom vrednošću signala takta **CRP** keš memorija **KEŠ** signalizira procesoru **CPU** da je na linijama $CDRRD_{7..0}$ validan podatak i da procesor **CPU** može da nastavi sa radom. Aktivna ili neaktivna vrednost signala **BUSY** signalizira procesoru **CPU** da je keš memorija **KEŠ** zauzeta ili ne, respektivno. Po startovanju operacije čitanja keš memorija **KEŠ** aktivnom vrednošću signala **PRQRD** postavlja signal **BUSY** na aktivnu vrednost. Ako ima saglasnosti očitani podatak se šalje procesoru **CPU**, signal **CRP** se postavlja na aktivnu vrednost, a signal **BUSY** na neaktivnu vrednost. Tako se omogućava procesoru da produži sa radom ili da startuje novu operaciju u keš memoriji. Ukoliko se dogodi da nema saglasnosti u keš memoriji **KEŠ**, iz memorije **MEM** se dovlači blok sa adresiranim podatkom, ali se najpre dovlači traženi podatak, pa tek onda ostatak bloka. Kada se dovuče traženi podatak, on se odmah šalje procesoru **CPU** i generiše aktivna vrednost signala **CRP**, kako bi se omogućilo da procesor ranije nastavi sa radom. Signal **BUSY** se zadržava na aktivnoj vrednosti, jer je keš memorija **KEŠ** i dalje zauzeta dovlačenjem ostatka bloka u deo za čuvanje podataka. Zbog toga se procesoru onemogućava da startuje novu operaciju sve dok svi bajtovi adresiranog bloka ne budu prebačeni iz memorije **MEM** u memoriju **KEŠ**.

Kod operacije upisa procesor **CPU**, pored toga što šalje adresu, na koju treba da se upiše podatak, po linijama $PAR_{15..0}$, po linijama $PDRWR_{7..0}$ šalje 8-bitni podatak koji se treba upisati na datu adresu i generiše aktivnu vrednost signala $PRQWR$ u trajanju jedne periode signala takta. Keš memorija **KEŠ** šalje procesoru **CPU** signale CRP i $BUSY$. Aktivna vrednost signala CRP signalizira procesoru **CPU** da može da nastavi sa daljim radom, dok aktivna vrednost signala $BUSY$ signalizira da je memorija **KEŠ** zauzeta, ukoliko se radi o neaktivnoj vrednosti, to je znak da memorija **KEŠ** nije zauzeta. Kod startovanja operacije upisa aktivnom vrednošću signala $PRQWR$ keš memorija **KEŠ** upisuje podatak sa linija $PDRWR_{7..0}$ u prihvatni registar podataka, generiše aktivnu vrednost signala CRP i signal $BUSY$ postavlja na aktivnu vrednost. Time se omogućava da procesor **CPU** nastavi sa radom, ali se i aktivnom vrednošću signala $BUSY$ sprečava da otpočne novu operaciju čitanja ili upisa. Signala $BUSY$ je aktivan sve dok se memorija **KEŠ** bavi upisom podatka iz prihvatnog registra u memoriju **MEM**, a ukoliko ima saglasnosti i u memoriju **KEŠ**.

Sa slike možemo da zaključimo da se procesor **CPU** sastoji iz:

- *operacione jedinice*
- *upravljačke jedinice*

Operaciona jedinica je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža za pamćenje binarnih reči, izvršavanje mikrooperacija na osnovu signala koji dolaze iz *upravljačke jedinice* i generisanje signala logičkih uslova koji 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 prema procesoru **CPU** i signala logičkih uslova.

2.2.1.1 Operaciona jedinica

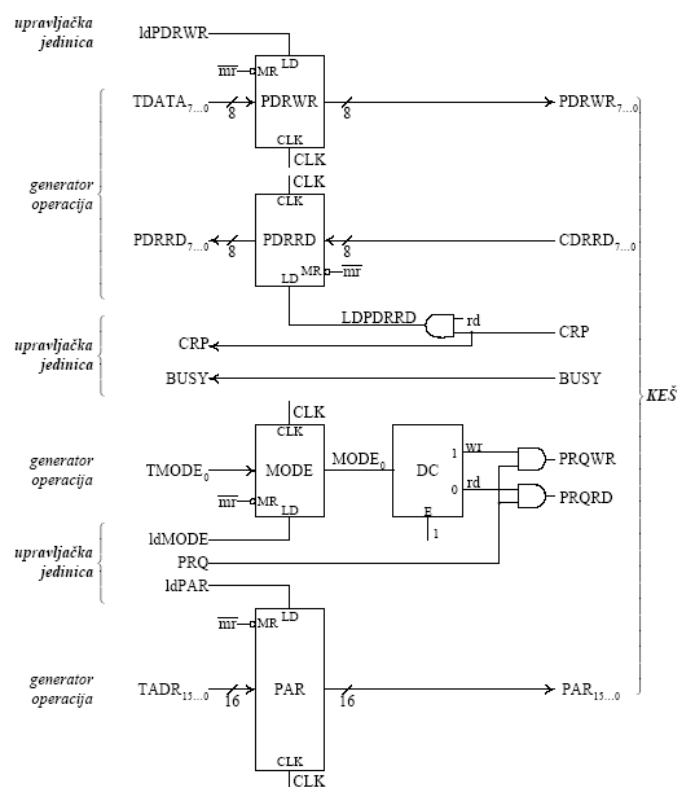
Operaciona jedinica sastoji se iz dva bloka:

- blok *keš interfejs*
- blok *generator operacija*

Blok *keš interfejs* služi za povezivanje procesora **CPU** i keš memorije **KEŠ**. Ovaj blok sadrži registre PAR , $PDRWR$, $PDRRD$, $MODE$, dekodera DC i dva logička I kola. Na slici su prikazani, način na koji je ostvareno povezivanje keš memorije i procesora, kontrolni signali (ld) upisa u registre, upravljački signal CRP , kao i način generisanja upravljačkih signala rd, wr , pomoću dekodera i upravljačkog signala PRQ .

Blok *generator operacija* sadrži brojače $ECNT$ i $WCNT$, flip-flop END i memoriju TAB . Brojač $ECNT$ služi za generisanje adresa ulaza u memoriju TAB , dok brojač $WCNT$ služi za realizaciju čekanja između dve operacije. Flip-flop END služi za indikaciju da se došlo do kraja generisanja svih ulaza TAB memorije.

Simulacija računarskog sistema sa asocijativnom keš memorijom



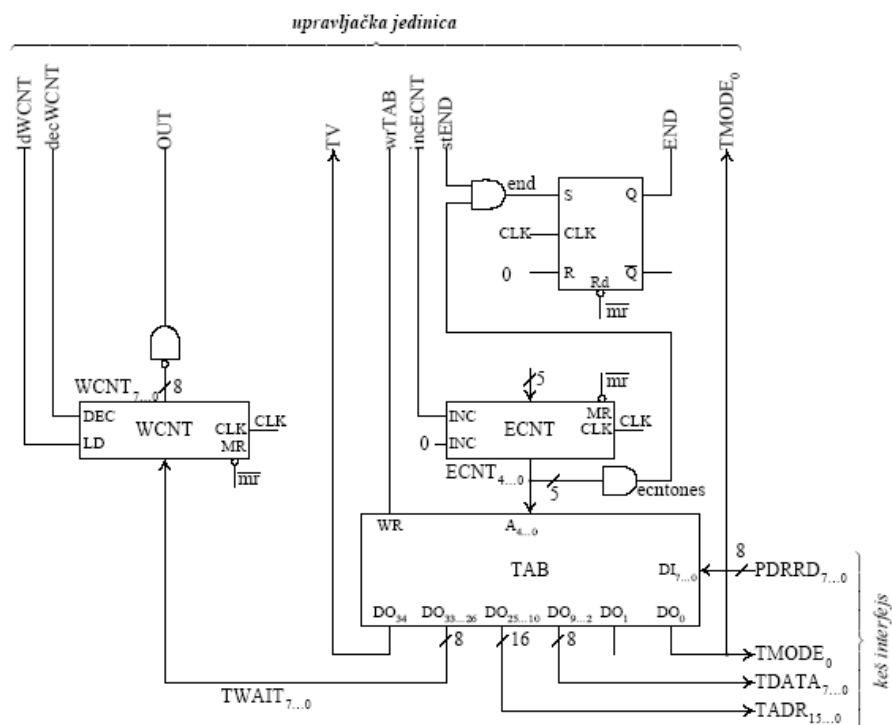
Izgled bloka keš interfejs

Memorija TAB ima 32 ulaza, što uzrokuje da se može generisati najviše 32 operacije keš memorije **KEŠ**. U svakom ulazu memorije TAB nalazi se informacija na osnovu koje se generiše jedna operacija keš memorije **KEŠ**.



Polje V ima funkciju da dati ulaz memorije TAB okarakteriše kao validan ($V=1$) ili ne ($V=0$), poljem WAIT definiše se koliko će taktiova procesor **CPU** između generisanja dve operacije, u polju ADR nalazi se adresa memorije **MEM** sa koje se podatak čita ili na koju se podatak upisuje. Polje DATA predstavlja podatak koji se upisuje u memoriju **MEM** u slučaju upisa ili podatak koji je očitao iz memorije **MEM** u slučaju operacije čitanja.

Polje MODE predstavlja kod operacije koju treba da realizuje keš memorija **KEŠ**. Vrednost 1 ukazuje da se radi o operaciji upisa, a vrednost 0 da se radi o operaciji čitanja.



Izgled bloka generator operacija

2.2.1.2 Upravljačka jedinica

Upravljačka jedinica u sklopu procesora **CPU** odgovorna je za sve upravljačke signale unutar samog procesora **CPU**, jer generiše signala koji kontrolišu blokove keš interfejs i generator operacija.

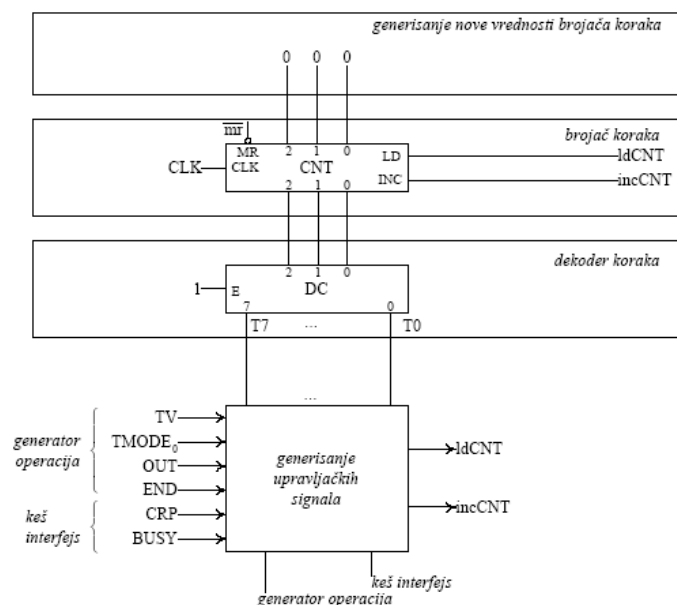
Upravljačka jedinica generiše i signale koje upravlja samom upravljačkom jedinicom.

Upravljačka jedinica sastoji se iz sledećih blokova:

- blok generisanje nove vrednosti brojača koraka
- blok brojača koraka
- blok dekođer koraka
- blok generisanje upravljačkih signala

Vidimo da se brojač koraka kontroliše sa dva upravljača signala *ldCNT* i *incCNT*. Oni dolaze iz bloka *generisanje upravljačkih signala*, i služe da se brojač nađe u režimu mirovanja, inkrementiranja ili režimu skoka. Na osnovu vrednosoti brojača, dekođer koraka genriše signale T_0, \dots, T_7 , koji se dalje koriste za generisanje upravljačkih signala.

Simulacija računarskog sistema sa asocijativnom keš memorijom



Izgled upravljačke jedinice bloka **CPU**

Upravljački signali bloka *keš interfejs* su:

- **ldPDRWR** — signal paralelnog upisa u registar PDRWR
- **ldMODE** — signal paralelnog upisa u registar MODE
- **ldPAR** — signal paralelnog upisa u registar PAR
- **PRQ** — signal startovanja jedne od dve operacije keš memorije

Upravljački signali bloka *generator operacija* 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
- **stEND** — signal upisa aktivne vrednosti u flip-flopa END

Upravljački signali za upravljanje samom *upravljačkom jedinicom* su:

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

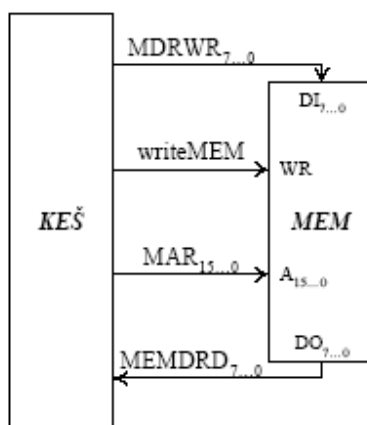
2.1.2 Memorija MEM

Memoriji **MEM** se obraća keš memorija **KEŠ** ili kada u okviru operacije upisa treba realizovati upise u memoriju **MEM** ili kada u okviru operacije čitanja u keš memoriji **KEŠ** nije otkrivena saglasnot, pa treba dovući blok iz memorije **MEM** i memoriju **KEŠ** i pročitati traženi podatak.

Adresa podatka koji treba da se pročita ili da se upiše u memoriju **MEM** šalje se po linijama MAR_{15..0}. Ukoliko se podatak čita da generisane adrese, signal writeMEM se drži na neaktivnoj vrednosti. U slučaju čitanja, očitani 8-bitni podatak se šalje po linijama

Simulacija računarskog sistema sa asocijativnom keš memorijom

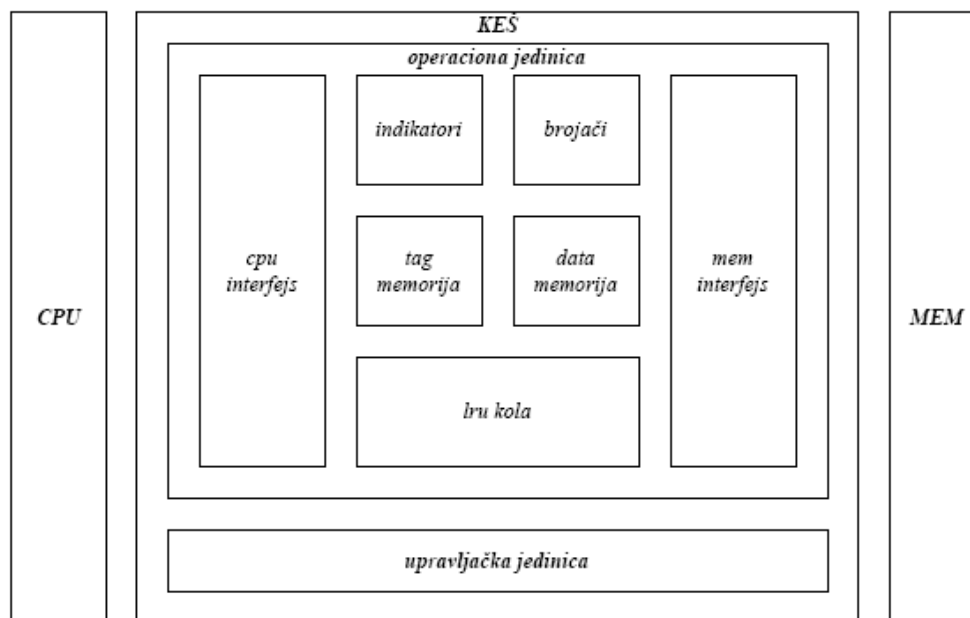
MEMDRD_{7..0}. Kod izvršavanja operacija upisa 8-bitni podatak koji treba upisati u memoriju **MEM** se šalje po linijama MDRWR_{7..0}, a signal writeMEM se drži na aktivnoj vrednosti. Uzeto je da vreme pristupa memoriji MEM zahteva deset perioda signala takta.



2.1.3 Keš memorija KEŠ

Keš memorija se sastoji od sledećih blokova:

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



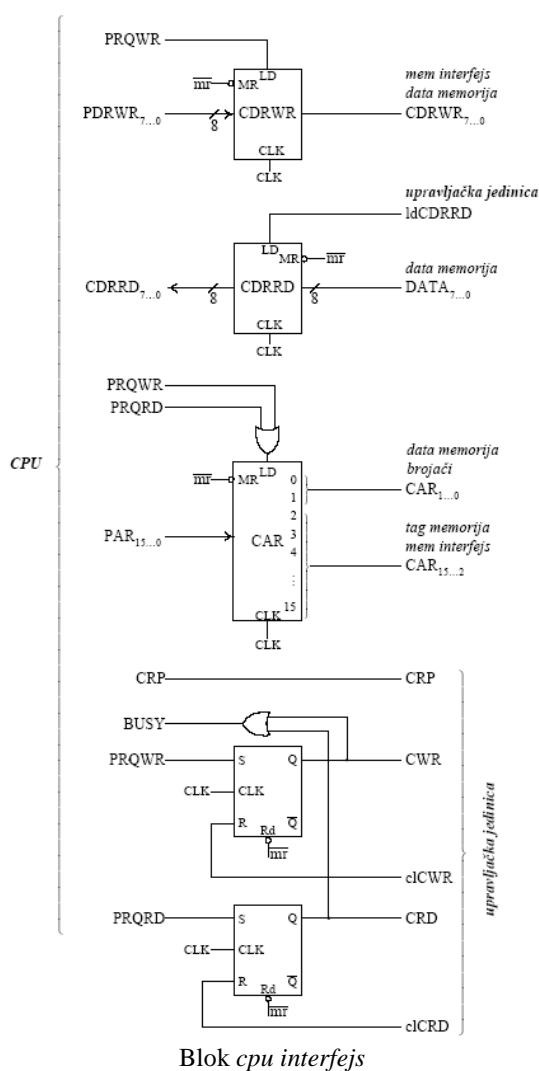
Operaciona jedinica služi za pamćenje podataka, izvršavanje operacija i prosleđivanje podataka ka memoriji MEM. Izvršavanje operacija keš memorije, realizovano je posredstvom upravljačkih signala upravljačke jedinice keš memorije.

2.1.3.1 Operaciona jedinica keš memorije

Operaciona jedinica keš memroje **KEŠ** sastoji se iz sledećih blokova:

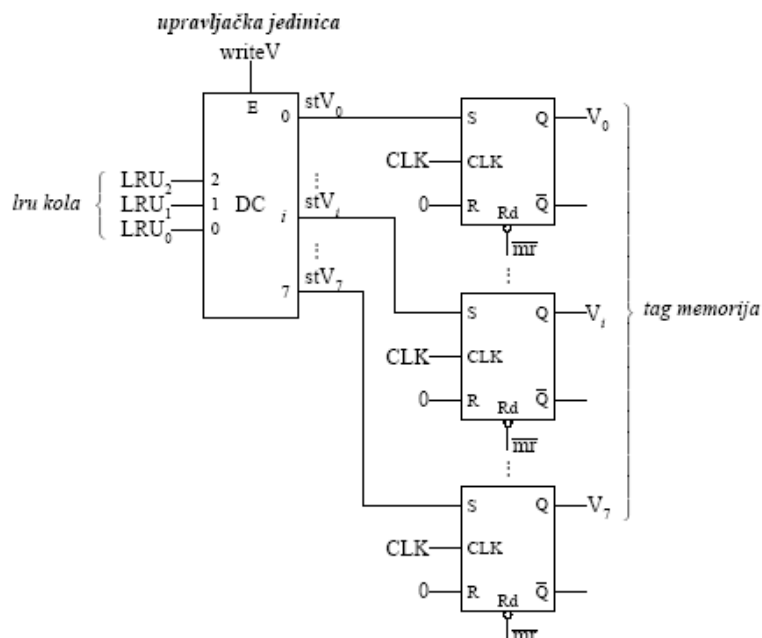
- blok *cpu interfejs*
- blok *indikator*
- blok *brojači*
- blok *tag memorija*
- blok *data memorija*
- blok *lru kola*
- blok *mem interfejs*

Blok *cpu interfejs* sadrži niz potrebnih registara i flip-floпова kako bi se realizovao prijem i prosleđivanje podataka, adresa, kao i razmena upravljačkih signala procesora **CPU** i keš memorije **KEŠ**.



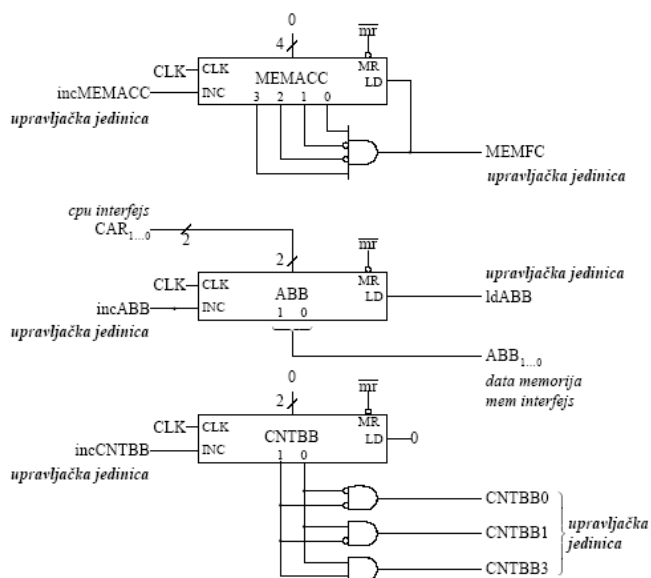
Simulacija računarskog sistema sa asocijativnom keš memorijom

Blok *indikator* služi za vođenje evidencije o validnosti ulaza memorije **KEŠ**. Za svaki od 8 ulaza potrebno je voditi ovu evidenciju. Ovaj blok sadrži flip-flopove V_0 do V_7 i dekorder DC.



Blok indikator

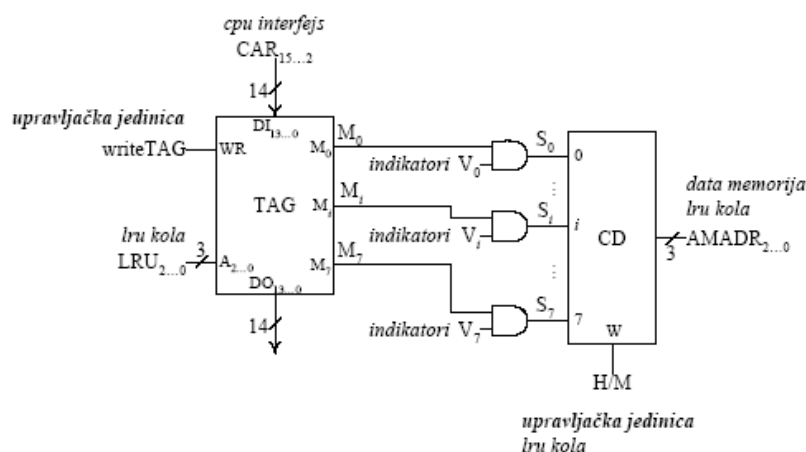
Blok *brojači* služi za generisanje adresa bajtova unutar bloka, za prebrojavanje broja prenetih bajtova iz memorije **MEM** u keš memoriju **KEŠ** i odbrojavanje onoliko perioda signala takta koliko je vreme pristupa memoriji **MEM**. U te svrhe koriste se tri brojača, i to: CNTBB (CuNter of Block Bytes), ABB (Address of Block Bytes) i MEMACC (MEMory ACCess time).



Blok brojači

Simulacija računarskog sistema sa asocijativnom keš memorijom

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**. Ovaj blok sastoji se od asocijativnog memorijskog modula TAG, koda CD i 8 logičkih I kola, koja se koriste za adresiranje ulaza kog koga je otkrivena saglasnost.



Blok *tag memorija*

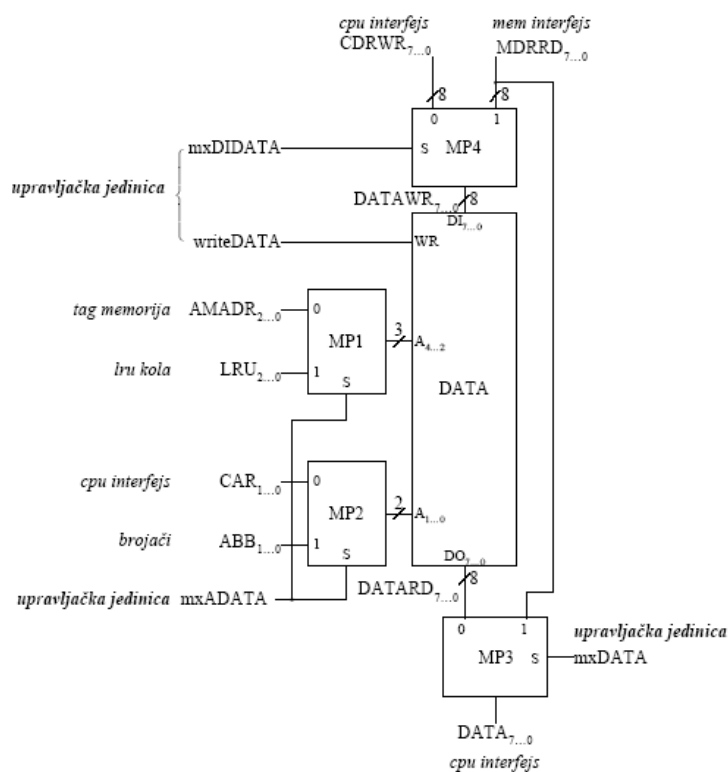
Blok *data memorija* služi za smeštanje 8 blokova podataka u keš memoriji **KEŠ**. Blok je realizovan pomoću memorijskog modula DATA koji je kapaciteta 32 bajta i potrebnog broja multipreksera.

Širina adrese modula DATA je 5 bita, niža dva bita adrese koriste se za adresiranje bajta u bloku koji se nalazi u modulu, a viših tri za određivanje broja bloka u operativnoj memoriji **MEM**. Multiplekseri se koriste za dobijanje adrese sa koje se sadržaj iz DATA modula čita ili na koju se upisuje, ako se radi o operaciji upisa.

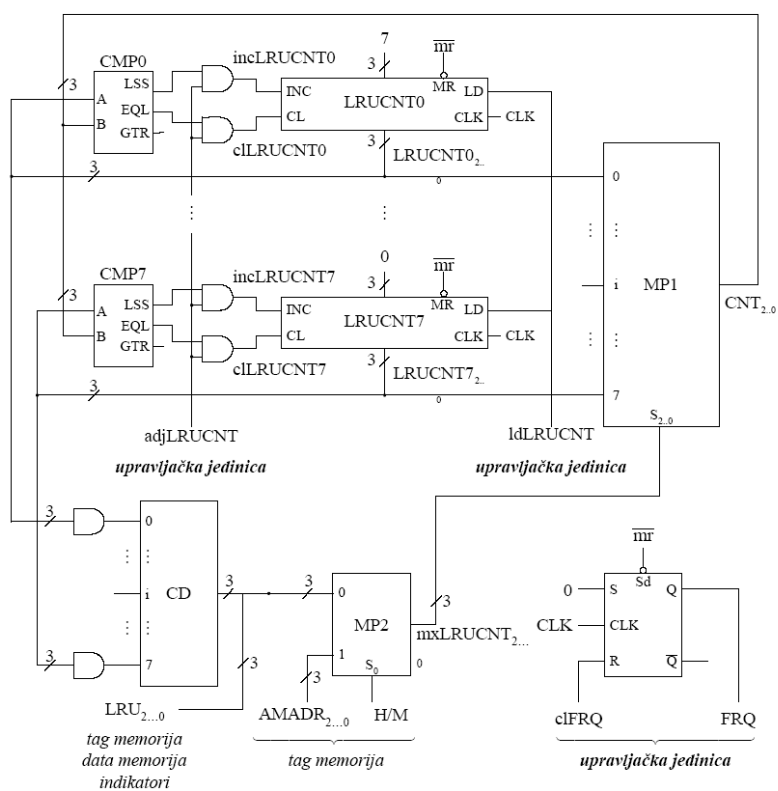
Blok *lru kola* koristi se za realizaciju LRU algoritma zamene. Ovaj blok sastoji se od 8 brojača po modulu 8, koliko i ima ulaza u keš memoriji **KEŠ**. Vrednosti na izlazima ovih brojača daju sliku o tome kome se ulazu koliko skoro pristupalo, a ova informacija potrebna nam je kod biranja ulaza za zamenu i kod biranja ulaza u koji se treba smestiti novi blok iz operativne memorije **MEM**. Pri svakom pristupu keš memoriji **KEŠ** sadržaji LRU brojača se ažuriraju. Pored brojača u ovom bloku nalazi se potreban broj komparatora, logičkih I kola, koder CD, multiplekseri kao i RS flip-flop, kojim se omogućava da se brojače upišu početne vrednosti, pri prvom generisanju bilo operacije čitanja bilo operacije upisa.

Blok *mem interfejs* sadrži registre MAR, MDRWR i MDRRD. Ovaj blok služi za povezivanje memorije **KEŠ** sa memorijom **MEM**. Registar MAR (Memory Address Register) služi za čuvanje adrese sa koje se podatak iz memorije **MEM** dovlači u memoriju **KEŠ** ili za čuvanje adrese na koju se podatak upisuje u memoriju **MEM**. Registar MDRWR (Memory Data Register for Write) služi za čuvanje podatka koji kod operacije upisa treba upisati u memoriju **MEM**. Registar MDRRD (Memory Data

Simulacija računarskog sistema sa asocijativnom keš memorijom



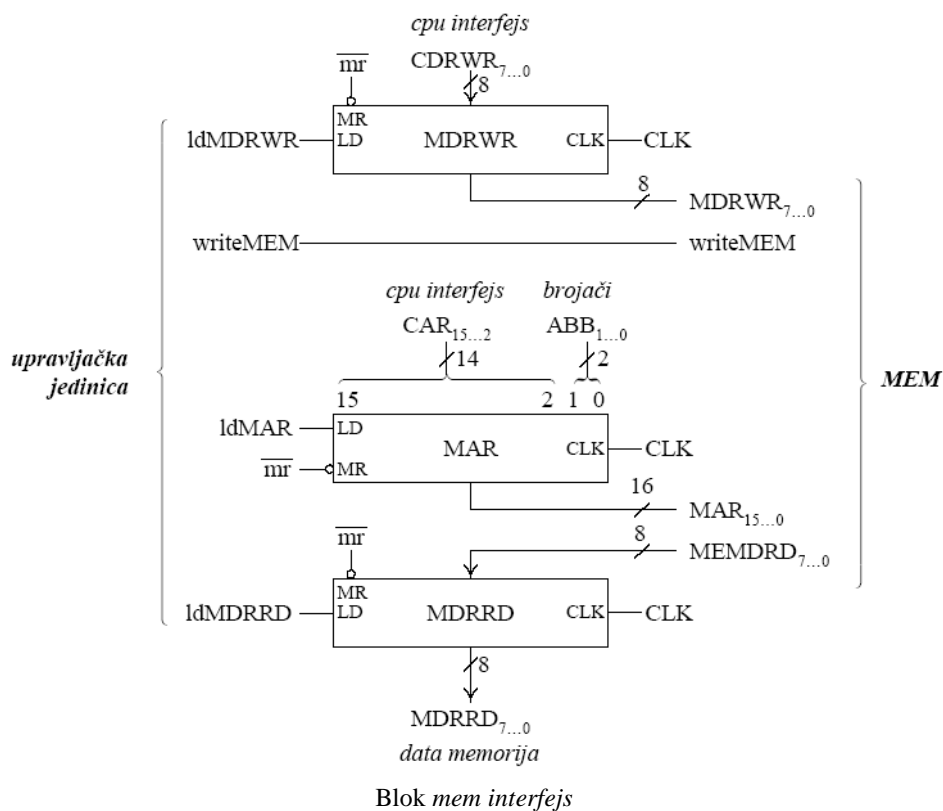
Blok data memorija



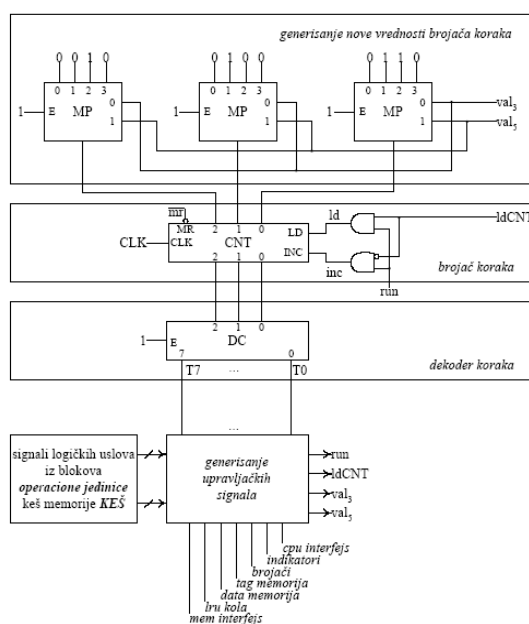
Blok lru kola

Simulacija računarskog sistema sa asocijativnom keš memorijom

Register for Read) služi za čuvanje podatka koji je kod operacije čitanja očitao iz memorije **MEM**.



2.2.3.2 Upravljačka jedinica



Izgled upravljačke jedinice bloka **KEŠ**

Simulacija računarskog sistema sa asocijativnom keš memorijom

Upravljačka jedinica keš memorije **KEŠ** služi za generisanje upravljačkih signala prema algoritmu asocijativnog preslikavanja. Ovim upravljačkim signalima vrši se upravljanje blokovima *operacione jedinice* keš memorije, kao i upravljanje signalima same *upravljačke jedinice*.

Upravljački signali upravljačke jedinice *ldCNT* i *run* omogućavaju da se brojač koraka pronađe u režimima mirovanju, inkrementiranju i režimu skoka. Signalima *val₃* i *val₅* se u režimu skoka određuje koja vrednost će biti propuštena kroz multipleksere (0, 3, 5) i upisana u brojač. Dekoder koraka formira diskretne korake formiranjem signala T_0, \dots, T_7 .

Upravljački signali bloka *cpu interfejs* 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
- **CRP** – signal komplementiranja operacije čitanja ili upisa

Upravljački signal bloka *indikator* je:

- **writeV** – signal upisa aktiven vrednosti u flip-flop V

Upravljački signali bloka *brojači* su:

- **incABB** – signal inkrementiranja sadržaja brojača ABB
- **incCNTBB** – signal inkrementiranja sadržaja brojača CNTBB
- **incMEMACC** – signal inkrementiranja sadržaja brojača MEMACC
- **ldABB** – signal paralelnog upisa u brojač ABB

Upravljački signal bloka *tag memorija* je:

- **writeTAG** – signal upisa u memorijski modul TAG

Upravljački signali bloka *data memorija* su:

- **mxDIDATA** – signal selekcije kroz multiplekser
- **mxADATA** – signal selekcije kroz multiplekser
- **writeDATA** – signal upisa u memorijski modul DATA
- **mxDATA** – signal selekcije kroz multiplekser

Upravljački signal bloka *lru kola* je:

- **adjLRUCNT** – signal ažuriranja sadržaja LRU brojača

Upravljački signali bloka *mem interfejs* su:

- **ldMAR** – signal paralelnog upisa u registar MAR
- **ldMDRWR** – signal paralelnog upisa u registar MDRWR
- **ldMDRRD** – signal paralelnog upisa u registar MDRRD
- **writeMEM** – signal upisa u memoriju MEM

Upravljački signali upravljačke jedinice:

Simulacija računarskog sistema sa asocijativnom keš memorijom

- **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 uslovim da je signal run aktivan, inkrementira vrednost brojača CNT
- **run** – signal čijom se aktivnom vrednošću omogućuje, u yavisnosti od vrednosti signala ldCNT, ili upisom nove vrednosti ili inkrementiranje vrednosti brojača CNT, a neaktivnom vrednošću, bez obzira na vrednost signala ldCNT, onemogućava promena vrednosti brojača CNT
- **val₃** i **val₅** – signali koji obezbeđuju generisanje vrednosti 0, 3, i 5 za upis u brojač CNT

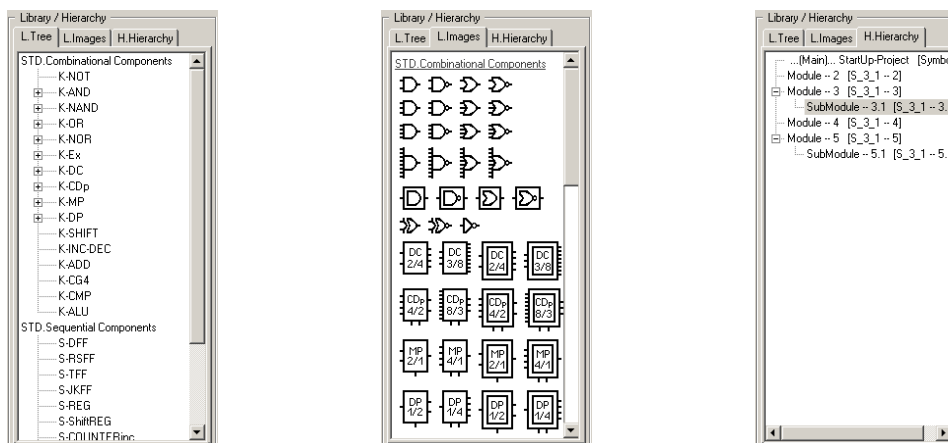
3 Opis softverskog alata IGoVSoEDS

Interaktivni generator vizuelnih simulatora elektronskih digitalnih struktura (IGoVSoEDS) je softverski alat za kreiranje i modifikaciju elektronskih digitalnih struktura. Alat obezbeđuje jednostavno kreiranje i upravljanje simulacijom, efikasan pregled stanja i vrednosti signala kreirane digitalne strukture, kao i rezultata simulacije. U daljem tekstu biće predstavljeni najkorišćeniji aspekti softverskog alata IGoVSoEDS, u ovom alatu realizovan je *simulator keš memorije sa asocijativnim preslikavanjem*.

3.1 Biblioteke kola i hijerarhijsko stablo modula

Alat IGoVSoEDS sadrži biblioteku kola koja se koriste za realizaciju kompleksnih digitalnih struktura. Ova biblioteka sadrži kombinaciona kola, sekvencijana kola, i specijalne module. Kod kombinacionih i sekvencijalnih kola postoje standardna kola kao i modifikovana kola.

Kada neko kolo želimo da postavimo na podlogu za prikaz, dovoljno je da levim tasterom miša kliknemo na kolo koje želimo da dodamo i ono će se pojaviti na podlozi. Kola možemo birati iz biblioteke L.Tree i L.Images. Na slici je prikazano kako ove biblioteke izgledaju, na poslednjoj slici, desno, prikazan je pregled hijerarhijske digitalne strukture.



Kao što možemo da vidimo na raspolagaju su nam:

- Standardna kombinaciona kola, sa označenim brojem ulaznih i izlaznih konektora: K-NOT, K-AND_i, K-NAND_i, K-OR_i, K-NOR_i, K-ExOR2, K-ExNOR2, K-DC_{k-m}, K-CDp_{k-m}, K-MP_{k-m}, K-DP_{k-m}, K-SHIFT, K-INC-DEC, K-ADD, K-CG4, K-CMP i K-ALU.
- Standardna kombinaciona kola, sa modifikovanim ulaznim konektorima: K-gAND, K-gNAND, K-gOR, K-gNOR, K-gDC_{k-m}, K-gCDp_{k-m}, K-gMP_{k-m} i K-

Simulacija računarskog sistema sa asocijativnom keš memorijom

gDP_{k-m}. Sva kola iz ove grupe imaju po jedan ulazni konektor na koji može da se poveže signal sa širinom od 1 do 32 bita, dok je izlazni signal uvek širine 1 bit.

- Standardna sekvencijalna kola: S-DFF, S-RSFF, S-TFF, S-JKFF, S-REG, S-ShiftREG, S-COUNTERinc i S-COUNTER.
- Specijalni moduli:
Sp-SubModule – *NonPort Module*, Sp-TriState – trostatički bafer koji omogućava povezivanje signala na magistralu, Sp-Generator – generator signala, Sp-Memory – memorijski modul.

Nakon pojave kola na podlozi za prikaz, kolo možemo pomerati uz pomoć funkcije *Move Object* iz menija za pomeranje (*Editor Mode Toolbar*) bilo gde u okviru same podloge.



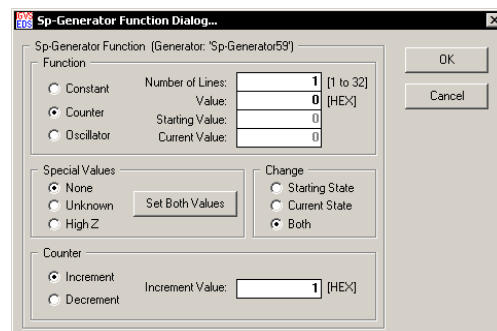
Jako korisna opcija je i promena širine konktora, uz pomoć funkcije *Expand Connectors*. O ovoj fukciji biće reči malo kasnije.

Ukoliko nam je u realizaciji digitalne strukture potrebno kolo kojeg nema u biblioteci alata, ono se može napraviti kombinacijom datih kola iz biblioteke.

3.2. Rad sa generatom signala i memorijskim modulima

Generator signala, *Sp-Generator*, može koristiti u tri režima rada:

- *Constant* – izlazni signal ne menja vrednost
- *Counter* – izlazni signal menja vrednost na način koji je definisan – povećanje (*Increment*), odnosno smanjivanje (*Decrement*) vrednosti izlaznog signala za vrednost polja *Increment Value*, odnosno *Decrement Value*, respektivno
- *Oscillator* – izlazni signal menja vrednost tako što se svi bitovi izlaznog signala invertuju



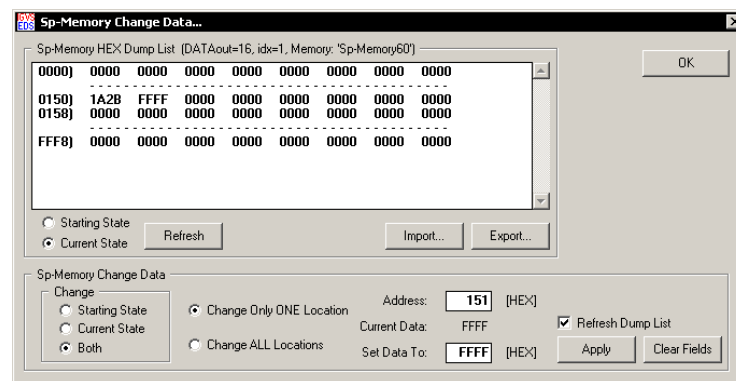
Simulacija računarskog sistema sa asocijativnom keš memorijom

Ukoliko je potrebno da se na neke linije ne dovodi ništa, potrebno je na te linije vezati Sp-Generator, konstantne vrednosti 0, što je u simulatoru dosta korišćeno.

Memorijski modul je jedan od posebnih modula koji se nalaze u alatu i kod ovog modula, linije za adresu imaju 16 bita, dok linije za podatke mogu biti širine 16, 8, 4, 2 i 1 bit.

Fukcije koje mogu biti izvršavane u radu sa memorijskim modulom su:

- *HEX Dump List...* – funkcija za otvaranje dijaloga za pregled sadržaja memorijskih lokacija memorijskog modula. Taster *Change* transformiše *HEX Dump List...* dijalog u *Change Data...* dijalog. Tasteri *Import...* i *Export...* pozivaju iste dijaloge kao i funkcije *Import Memory Data...* i *Export Memory Data...*, respektivno.
- *Change Data...* – funkcija za otvaranje dijaloga za pregled i modifikaciju sadržaja memorijskih lokacija memorijskog modula.
- *Import Memory Data...* – dijalog za učitavanje sadržaja memorijskih lokacija iz datoteke.
- *Export Memory Data...* – dijalog za snimanje sadržaja memorijskih lokacija u datoteku.



3.3. Promena širine konektorima (*Expand Connectors...*)

Dati skup kola koje možemo naći u biblioteci moguće je proširiti na veći broj kola korićenjem funkcije *Object/Expand Connectors...* Funkcija omogućava promenu širine signala na konektorima od 1 do 32 bita, sa korakom od 1 bita ili više, zavisno od toga kom modulu se menja širina signala na konektorima. Ovom funkcijom moguće je promeniti širinu signala na konektorima kod svih kola koja imaju portove, a nemaju povezane signale. Nakon povezivanja kola, nije moguće izvršiti promenu širine konektora.

Simulacija računarskog sistema sa asocijativnom keš memorijom

Funkciji se pristupa pomoću menija *Object*, a zatim biramo iz padajućeg menija opciju *Expand Connectors...*

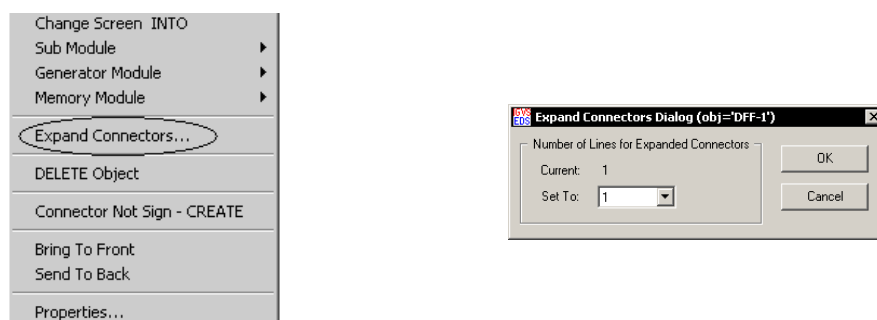
Postoje dve grupe koje se koriste u radu sa funkcijom *Expand Connectors* i to:

- kola kod kojih se širina signala menja na isti način na svim konektorima
- kola kod kojih se širina signala ne menja na isti način na svim konektorima

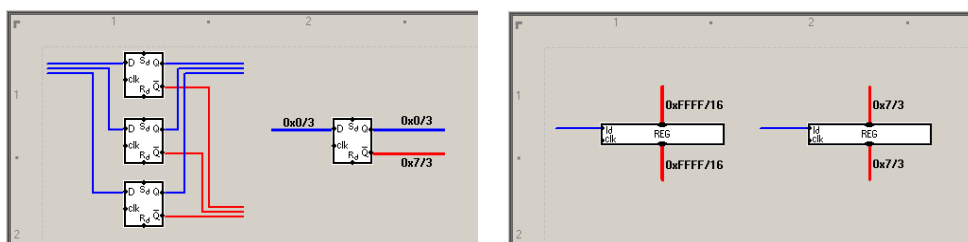
Kola iz prve grupe, nakon izbora iz biblioteke, imaju sve konekcije širine 1 bit. Korišćenjem funkcije *Expand Connectors* menja se širina svih konektora na objektu. Objekti iz druge grupe, nakon izbora iz biblioteke, imaju dve vrste konektora, širine 1 bita i širine 16 bita. Korišćenjem funkcije *Expand Connectors* menja se jedino širina onoga dela sa 16 bita, dok konekcija sa 1 bitom ostaje fiksna.

U slučaju da koristimo memorijski modul, nije moguće menjati širinu signala na konektorima sa korakom 1 bit, već možemo izabrati samo jednu od dozvoljenih vrednosti, a to su: 8, 4, 2, 1 dok je vrednost 16 podrazumevana. Promenom širine ulaznih i izlaznih linija kod memorijskog modula zapravo menjamo širinu memorijske reči samog modula. Ulazne linije za adresu imaju 16 bita.

Prikaz rada sa funkcijom *Expand Connectors...*



Prikaz delovanja funkcije *Expand Connectors...*



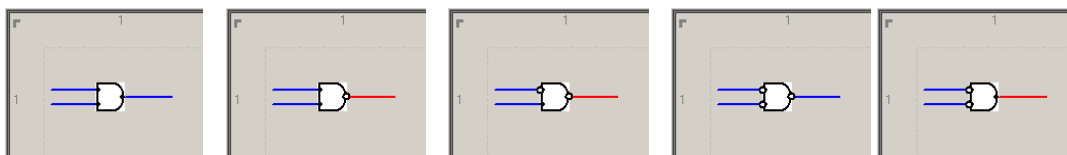
3.4 Invertori na konektorima kola (*Object/Connector Not Sign*)

Funkcijom *Connector Not Sign* iz menija *Object* vrši se dodavanje kružića na konektorima, tj. *not* znaka, čime se postiže da vrednost signala koji je povezan na dati

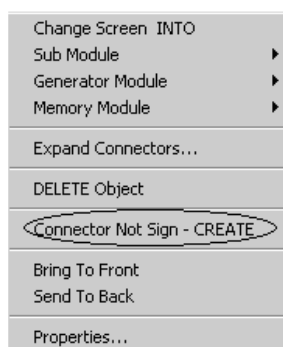
Simulacija računarskog sistema sa asocijativnom keš memorijom

konektor bude invertovana. Funkcija nije dostupna za specijalni modul *Sp-Generator*, kao i za *Port Module* i *NonPort Module* objekte. Funkcija se aktivira otvaranjem padajućeg menija za objekte i odnosi se na konektor najbliži na slici položaju miša u trenutku padajućeg menija. Opcije CREATE i REMOVE su dostupne prilikom aktiviranja ove funkcije, i to ukoliko kružić ne postoji ili kružić postoji, respektivno.

Delovanje funkcije *Object/Connector Not Sign*:

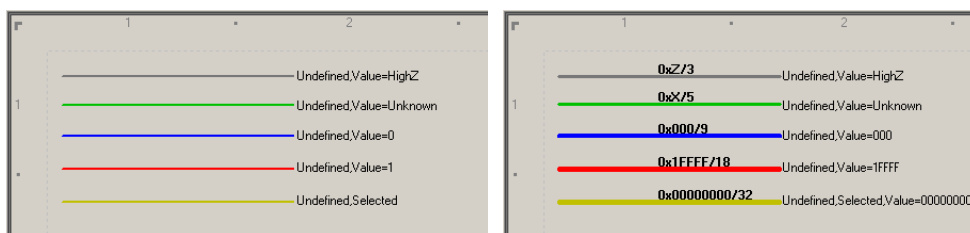


Rad sa funkcijom *Object/Connector Not Sign*:



3.5 Prikaz linija signala i njihovo kreiranje i modifikacija

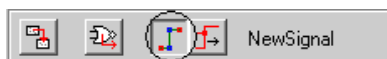
Linije signala mogu biti prave ili izlomljene linije čiji se krajevi nalaze na konektorima kola, na portovima *Port Module* objekata, ivičnim linijama *NonPort Module* objekata, postojećim linijama signala, ali mogu se naći i na slobodnom prostoru podloge za prikaz digitalnih struktura. Uloga linija signala nije samo da predstavljaju signale u šemi nego i da služe za praćenje rezultata simulacije i prikazivanje broja bitova signala.



U zavisnosti o vrednosti na samoj liniji, stanje visoke impedanse, nepozната vrednost, stanje logičke nule ili jedinice i selektovani signal u trenutku rada, ona može biti u nekoj boji. Na slici (gore) su prikazane vrednosti koje se mogu naći na linijama signala kao i prikaz širine linija, tj. debljina linija signala.

Simulacija računarskog sistema sa asocijativnom keš memorijom

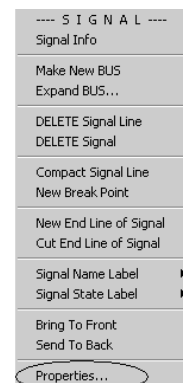
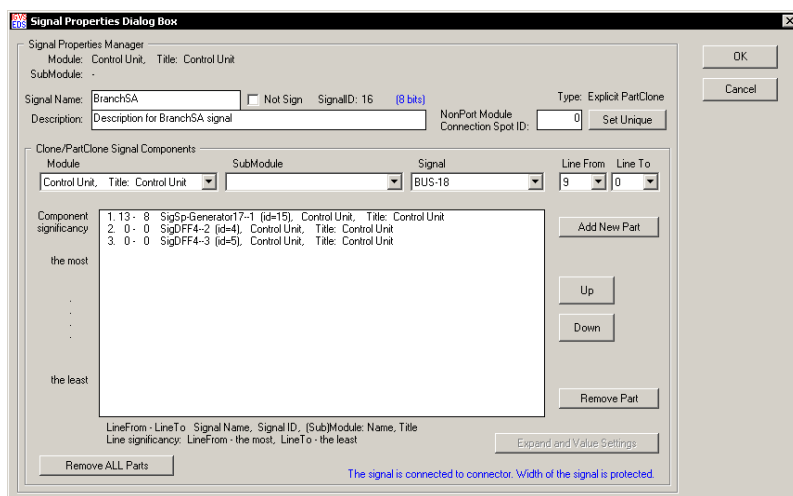
Signal postavljamo na podlogu za prikaz pomoću stavke *New Signal*, a postavljeni signam na podlozi možemo pomerati stavkom *Move Signal*. Obe stavke nalaze se u *Editor Mode Toolbar*-u.



Krajnje i prelomne tačke signala označene su crvenom, žutom i plavom bojom. Dvostrukim aktiviranjem levog klika miša određuje se početna ili prelomna tačka, dok se dvostrukim desnim klikom ili tasterom F9 završava kreiranje linije. Kod crtanja signala koji treba da vodi do konektora ili porta, jako je korisno crtati prvo početni tačku, a zatim označiti krajnju tačku na samom konektoru ili portu modula, jer na taj način se ne mora koristiti eksplicitno označavanje krajnje tačke, već alat sam završava crtanje linije, a pored toga izbegava se mogućnost nepovezivanja, do koje može doći ukoliko se krajnja tačka linije označava desnim dvoklikom ili tasterom F9. Ponekad baš zbog ovakvih neopreznosti možemo da steknemo privid da je signala povezan na strukturu, a da zapravo nije.

Kada se neki signal sastoji od više linija, tj. bitova koji dolaze iz različitih izvora, neophodno je navesti delove same linije i to u tačnom redosledu, od bita sa najvećom važnošću do bita na poziciji 0. Ovo nam omogućava funkcija *PartCloneSignal*. Delove možemo dodavati kada selektujemo signal kome dodeljujemo tačno od kojih se signala sastoji, a zatim izaberemo opciju *Properties*. Nakon ove iteracije, možemo dodavati delove signala ili jednostavno vršiti dodeljivanje početne vrednosti, ako je to moguće, tj. ako je signal ulazni za datu strukturu.

Na slici je prikazano kreiranje *Clone/Part Clone* signala i dijalog za rad sa selektovanim signalom.



Simulacija računarskog sistema sa asocijativnom keš memorijom

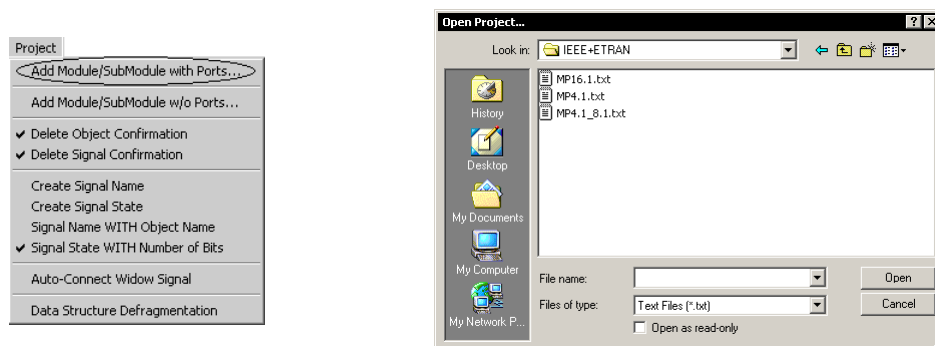
Signalu se na ovaj način, pokretanjem opcije *Properties* iz padajućeg menija, može menjati naziv (*Signal Name*), opis (*Description*), ispisivati komplement (*not Sign*), dodeljivati jedinstveni broj u sistemu numeracije signala za potrebe rada funkcije *Replace With ALL Connections...* (*Set Unique*). Za tipove signala *Undefined*, *Clone* i *PartClone* dostupan je i donji deo dijaloga *Signal Properties Dialog* (*Clone/PartClone Signal Components*). Obezbeđen je sistem sa listama modula (*Module*), podređenih modula (*SubModule*) i signala (*Signal*), preko kojih je dostupan svaki signal koji je vidljiv na modulu koji je aktivan prilikom otvaranja padajućeg menija *Signal Menu*.

Ako želimo da signalu dodamo deo (*Add New Part*), potrebno je izabrati modul iz kog biramo signal, zatim izabrati podmodul ako je potrebno (*SubModule*), a zatim izabrati signal koji će da čini jedan deo ili ceo signal koji smo selektovali i čiju opciju *Properties* koristimo. Na kraju biramo koje linije će činiti deo selektovanog signala (*Line From/Line To*) i aktiviranjem tastera *Add New Part*, odabrane linije signala postaju komponente kojima se kreira zavišni signal. Naravno, već je napomenuto da se strogo mora voditi računa da se linije koje čine deo linija navode u tačnom redosledu, od najbitnije do najmanje bitne. Ako je neki signal isto što i neki drugi u digitalnoj strukturi, tj. nasleđuje sve linije nekog signala, kažemo da je taj signal *Clone Signal*, u svim drugim situacijama kažemo da se radi o *PartClone Signal*-u.

Ponekad je potrebno da se signalu dodeli početna vrednost, u slučajevima kad on ne nasleđuje vrednosti nekog od signala u digitalnoj strukturi ili nije povezan na neki konektor ili port. Tada je potrebno koristiti funkciju *Value Settings*.

3.6 Dodavanje modula u drugu elektronsku digitalnu strukturu

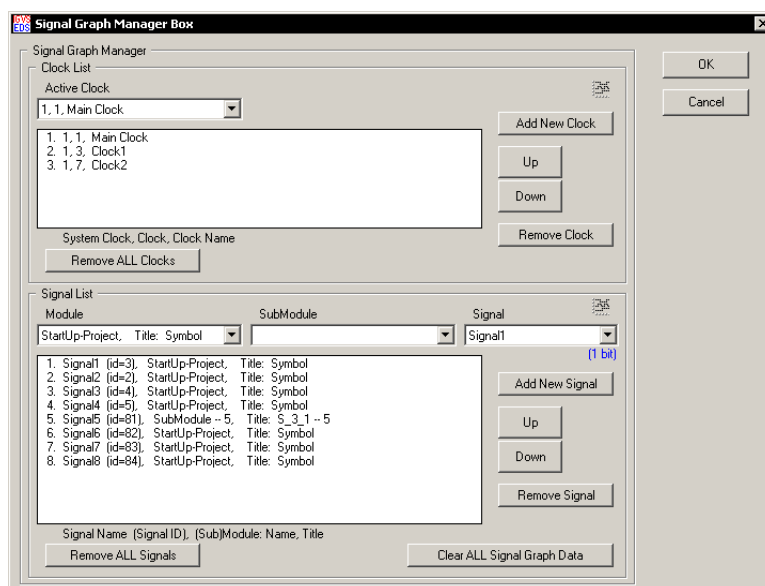
Dodavanje podmodula u drugu elektronsku strukturu ostvaruje se biranjem opcije *Add Module/SubModule with Ports...* ili opcije *Add Module/SubModule w/o Ports...* iz menija *Project*. Ovim opcijama dodaje se modul sa i bez portova, respektivno. Ukoliko kod kreiranja nekog modula ne definišemo njegove portove, on će se kasnije u digitalnoj strukturi, automatski pojavljivati kao modul bez portova.



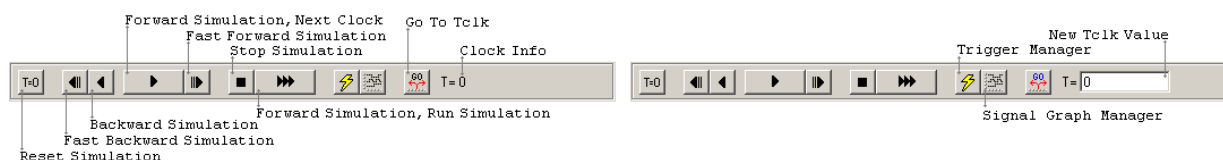
3.7 Rad sa parametrima simulacije i upravljanje simulatorom

Rad sa parametrima simulacije podrazumeva podešavanje vrednosti na osnovu kojih ćemo moći da pratimo ostale vrednosti, kao rezultate rada date digitalne strukture. Da bismo lako mogli da posmatramo da li se simulacija odvija po planu, možemo definisati signale koji će se naći u listi signala. Tako postizemo brz pregled trenutnog stanja linija.

Na slici je prikazan dijalog za rad sa listom signala za pregled promena vremenskih oblika signala (*Signal Graph Manager*), koji se otvara aktiviranjem tastera grafičkog skupa funkcija *Signal Graph*. Ukoliko želimo da promenimo redosleg signala u listi, to je moguće postići pritiskom na *Up* i *Down* polja.



Upravljanje simulacijom postiže se korišćenjem grafičkog skupa funkcija *Simulation Toolbar*. Izgled je dat na slici.



Tastera za upravljanje radom simulacije obezbeđuje funkcije za:

- postavljanje početnog stanja simulatora, odnosno stanja T_0 , za $T_{\max}=0$ (*Reset Simulation*)
- postavljanje simulatora na prethodno dostignute vremenske trenutke, čime se pokreće *RePlay* režim rada simulatora (*Fast Backward Simulation* i *Backward Simulation*)

Simulacija računarskog sistema sa asocijativnom keš memorijom

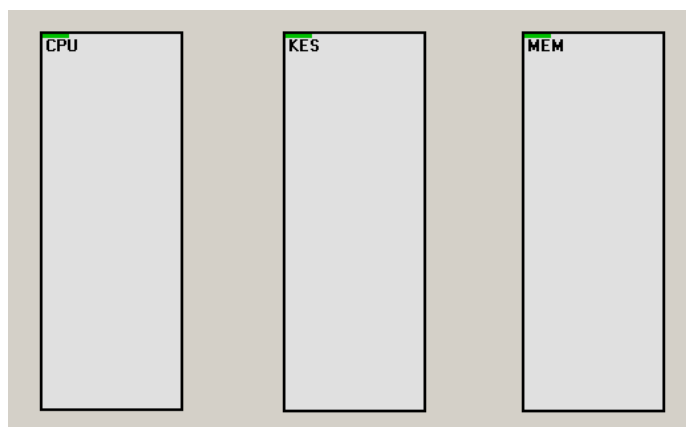
- pokretanje simulatora za jedan takt unapred (*Forward Simulation, Next Clock*)
- postavljanje simulatora na vremenske trenutke koji se nalaze iza vremenskih trenutaka koji se posmatraju u *RePlay* režimu rada simulatora (*Fast Forward Simulation*)
- zaustavljanje simulacije (*Stop Simulation*)
- pokretanje simulatora za više taktova unapred (*Forward Simulation, Run Simulation*)
- postavljanje simulatora na proizvoljan vremenski trenutak pre ili iza trenutno dostignutog vremenskog trenutka simulacije T_{\max} (*Go To Tclk*).

Za potrebe efikasnijeg kreiranja i modifikacije veoma složenih digitalnih struktura, realizovan je parametar *ReConfiguration Always ON*. Ukoliko ovaj parametar nije potvrđen, sa promenom digitalne strukture ne ažurira se struktura podataka koja čuva opis strukture i ne ažurira se stanje simulatora. U ovom režimu rada, skup funkcija za upravljanje radom simulacije nije dostupan, i pokriven je porukom *ReConfiguration is Stopped. Some values likely are incorrect*.

4 Prikaz keš memorije sa asocijativnim preslikavanjem realizovane alatom IGoVSoDES

Dati računarski sistem sastoji se od tri modula koji u sebi sadrže podmodule. To su moduli: procesor **CPU**, keš memorija **KEŠ** i memorija **MEM**. U procesoru se generišu operacije koje treba izvršiti, keš memorija preuzima ove operacije, a zatim ih izvršava i vrši pristup operativnoj memoriji ukoliko je to potrebno. Procesor prihvata očitane podatke preko linija CDRRD, prima potvrdu o završenoj operaciji preko linije CRP i signal da je keš memorija zauzeta preko linije BUSY. Preko linija PAR, PDRWR, PRQRD, PRQWR keš memoriji šalje adresu operativne memorije sa koje je potrebno očitati podatak ili na koju je potrebno upisati podatak, bajt podatak koji treba upisati na željenu adresu, da li se radi o zahtevu za čitanje ili zahtevu za upis, respektivno.

Keš memorija komunicira sa operativnom memorijom signalima MDRWR, writeMEM, MAR, MEMDRD i to preko linija MDRWR šalje podatak koji je potrebno upisati na adresu specificiranu sadržajem linija MAR. Po linijama MEMDRD šalje se podatak očitani sa adrese specificirane takođe linijama MAR, dok se linijom writeMEM šalje signal upisa u memoriju MEM. Svi signali sem signala MEMDRD, koji je ulazni, su izlazni za modul KEŠ.

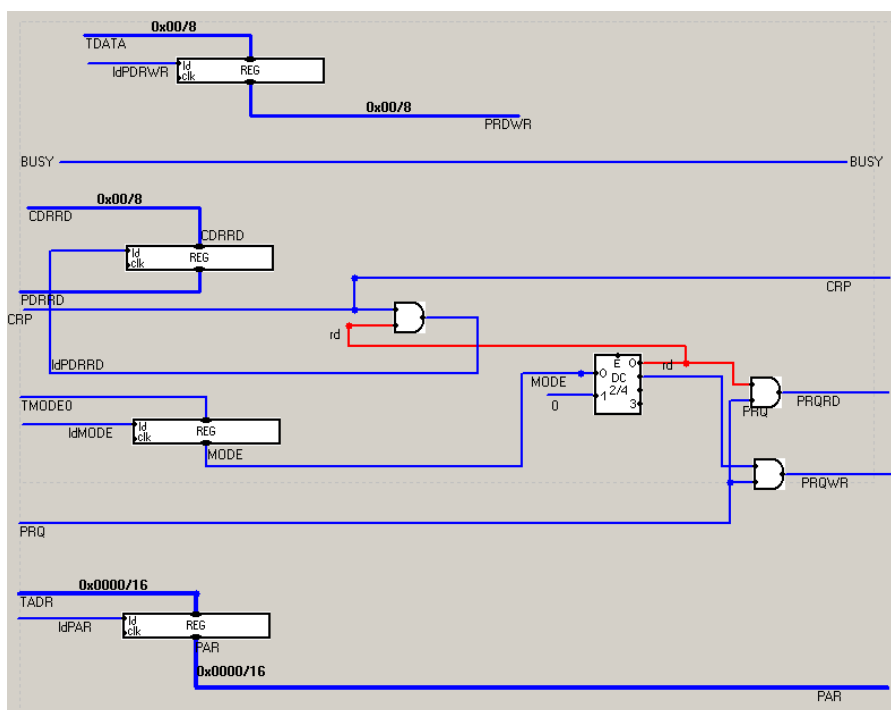


4.1 Procesor CPU

4.1.1 Blok keš interfejs

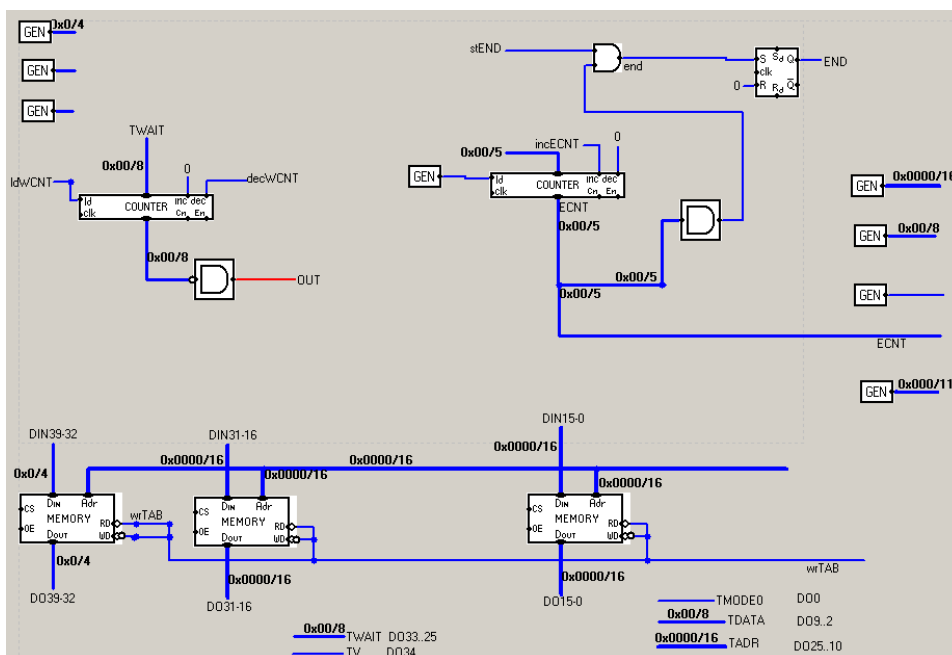
Blok **CPU keš interfejs** realizovan je prema slicisa strane 26 iz glave 2.2.1.1. Ovaj modul služi za povezivanje i komunikaciju modula **KEŠ** i modula **CPU**, a sastoji se od četiri registra PDRWR, PDRRD, MODE i PAR, kao i logičkih kola koje koristimo za formiranje upravljačkih signala za čitanje i upis.

Simulacija računarskog sistema sa asocijativnom keš memorijom



4.1.2 Blok generator operacija

Blok *CPU generator operacija* realizovan je prema slici na strani 27 iz glave 2.2.1.1. Posmatrani modul koristi se za generisanje operacija tako što preko svog ECNT brojača, adresira sekvencijano ulaze TAB memorije. Ulazi memorije TAB su širine 35 bita i



Simulacija računarskog sistema sa asocijativnom keš memorijom

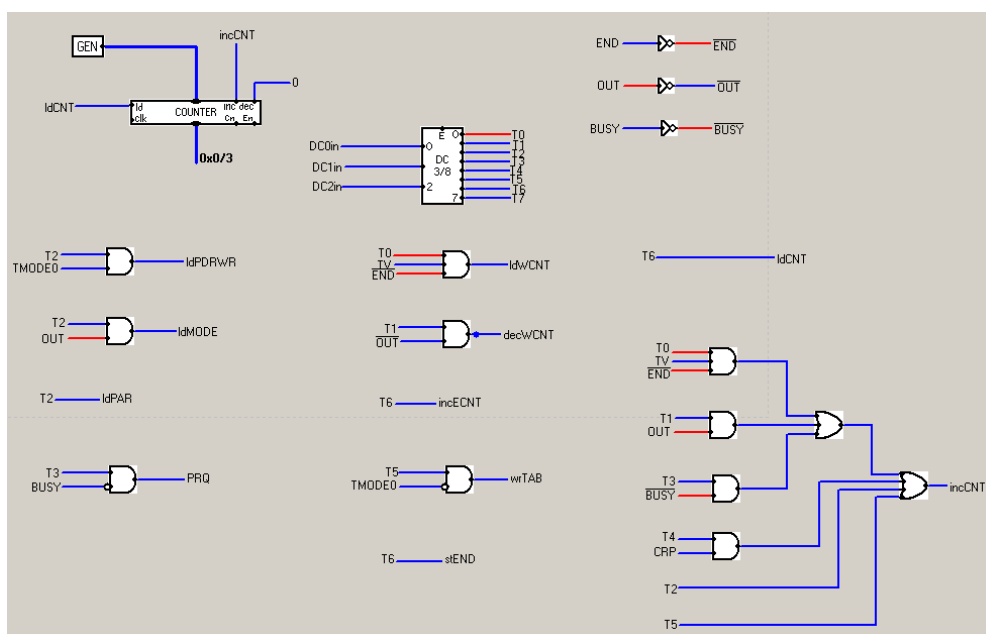
prosleđuju izlaze TV, TWAIT, TADR, TDATA, TMODE. Pre početka simulacije potrebno je ručno uneti sadržaj memorije TAB, odnosno operacije koje bi trebalo da se izvršavaju kada simulacija počne.

Posmatrani izlazi služe za dalje upravljanje realizovanim keš sistemom. Značenje i uloga portova izlaznih signala objašnjena je na strani 25.

Modul TAB memorije, prikazan je kao tri memorijska modula, poseduje tri ulazna porta, A_{4..0}, wrTAB i PDRRD, koji služe za adresiranje linija TAB memorije i upis očitanoj sadržaja sa linija PDRRD. Signale TV, END, TMODE0 i OUT koristi upravljačka jedinica procesora u svom kreiranju signala kojima se upravlja modulom **CPU**, kao i njegovim podmodulima.

4.1.3 Blok CPU upravljačka jedinica

Blok **CPU upravljačka jedinica** ima ulogu da generiše upravljače signale za blokove *keš interfejs* i *generator operacija* kao i za samu upravljačku jedinicu. Ovaj blok realizovan je prema slici iz glave 2.2.1.2, strana 28.

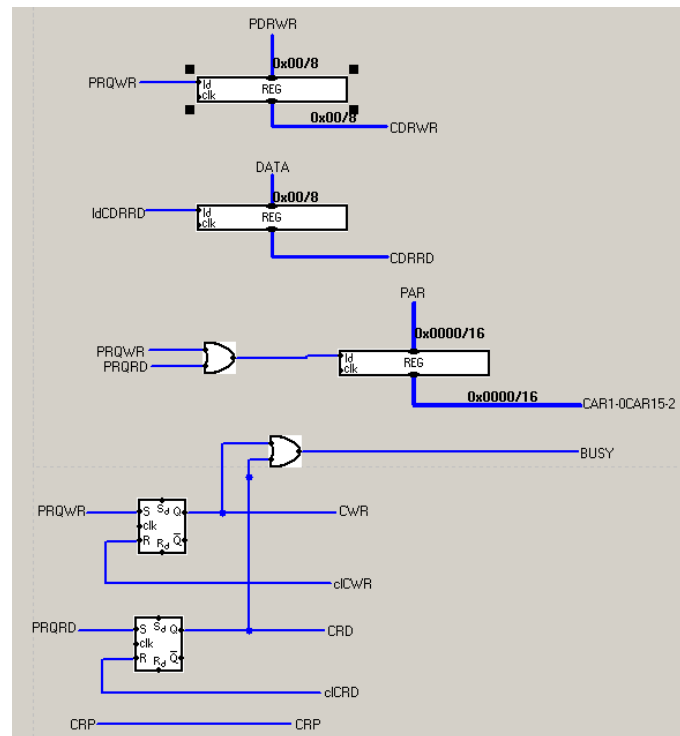


4.2 Keš memorija KEŠ

4.2.1 CPU interfejs

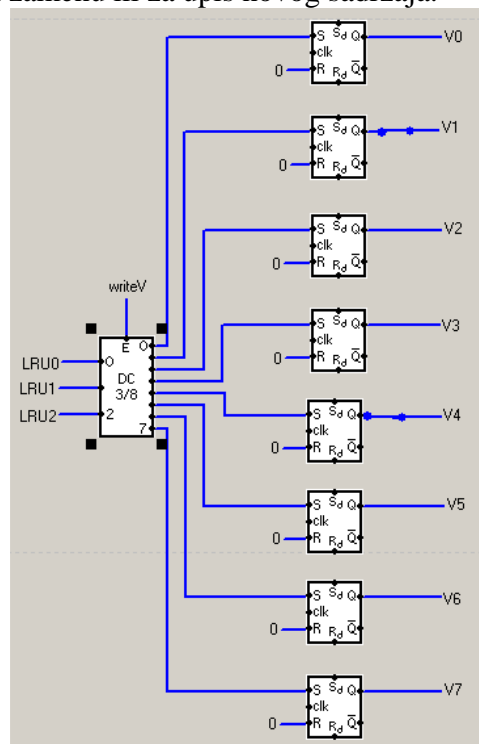
Blok *cpu interfejs* služi za komunikaciju između modula **CPU** i modula **KEŠ**. On prima podatke i signale od procesora **CPU** i prosleđuje je ih ostalim blokovima modula **KEŠ**. Realizacija ovog modula izvršena je prema slici iz glave 2.1.3.1 strana 30.

Simulacija računarskog sistema sa asocijativnom keš memorijom



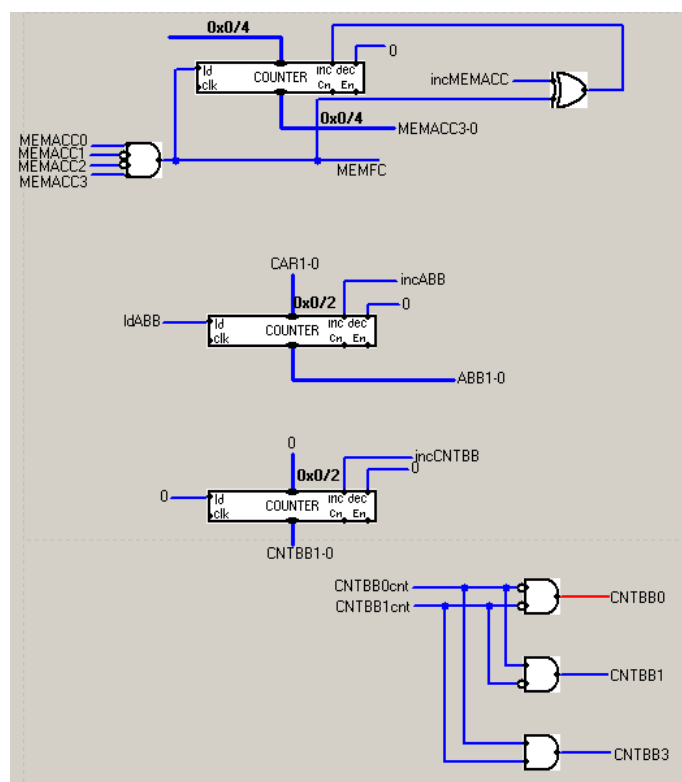
4.2.2 Blok indikatori

Ovaj blok koristi se za evidenciju zauzetosti ulaza memorije TAG, što je jako bitno kod određivanja ulaza za zamenu ili za upis novog sadržaja.



4.2.3 Blok brojači

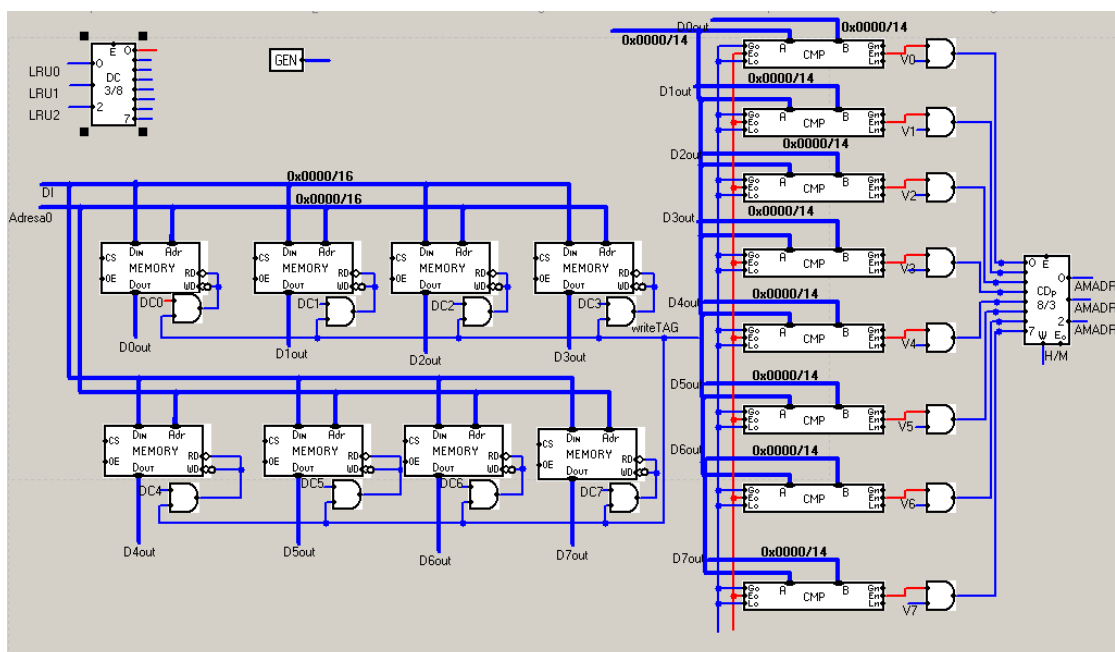
Uloga ovog bloka opisana je u glavi 2.1.3.1 i sam modul realizovan je prema slici sa strane 31. Izlaze ovog modula koristi *upravljačka jedinica*, *data memorija* i blok *mem interfejs*. Izvršena je mala dopuna šeme, kod inkrementiranja brojača MEMACC, jer je nemoguće učitati vrednost sa ulaza kada je aktivna i vrednost na konektoru *inc*.



4.2.4 Blok tag memorija

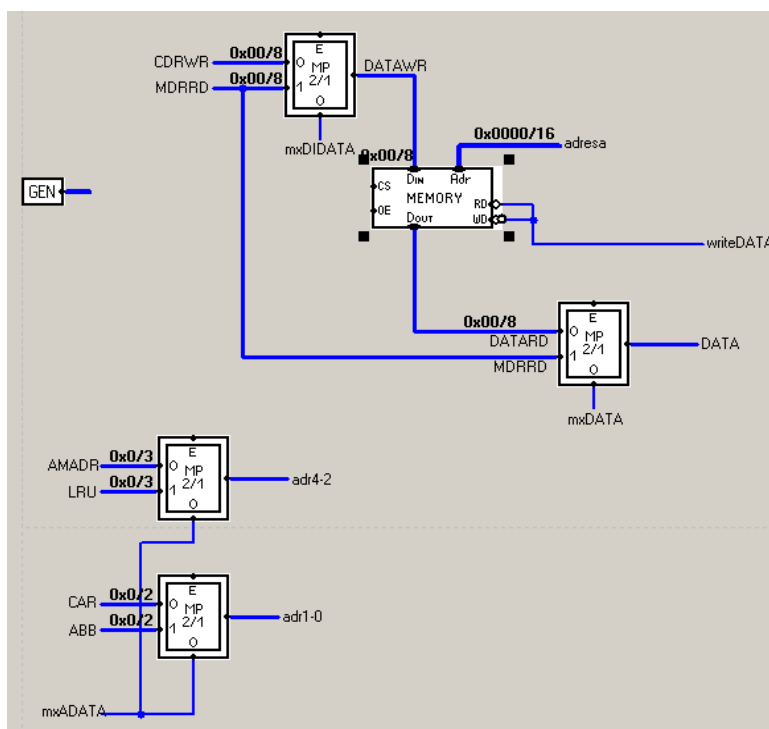
Blok *tag memorija* sadrži asocijativni memorijski modul i realizovana je po ugledu na šemu sa strane 32. Problem je bio kako realizovati memorijski modul koji je asocijativan, ako su dati samo standardni RAM moduli. Asocijativna memorija zahteva komplikovanu logiku za dobijanje signala Match. Zbog toga je ovaj modul realizovan tako sto je za svaki ulaz memorije TAG korišten po jedan memorijski modul, kako bi se postiglo da se sadržaj svakog ulaza čuva zasebno od ostalih. Na ovakav sistem primenjeno je upoređivanje sa signalom koji dolazi po linijama DI i tako je ostvareno formiranje signala Match. Ukoliko dođe do poklapanja sa sadržajem nekog ulaza memorije TAG, znači da se adresirani sadržaj već nalazi u keš memoriji, pa se na ulaz sa istim brojem kodera šalje 1. Dalje izlazi kodera koriste se u drugim delovima modula keš memorije **KEŠ**.

Simulacija računarskog sistema sa asocijativnom keš memorijom



4.2.5 Blok data memorija

Blok *data memorija* čuva sadržaje blokova čiji su brojevi zapisani u ulazima memorije TAG. Prilikom modifikacije podatka čiji se blok nalazi u keš memoriji, modifikuje se i sadržaj memorije DATA. Realizacija je sprevedena na osnovu slike iz glave 2.1.3.1 sa strane 33.

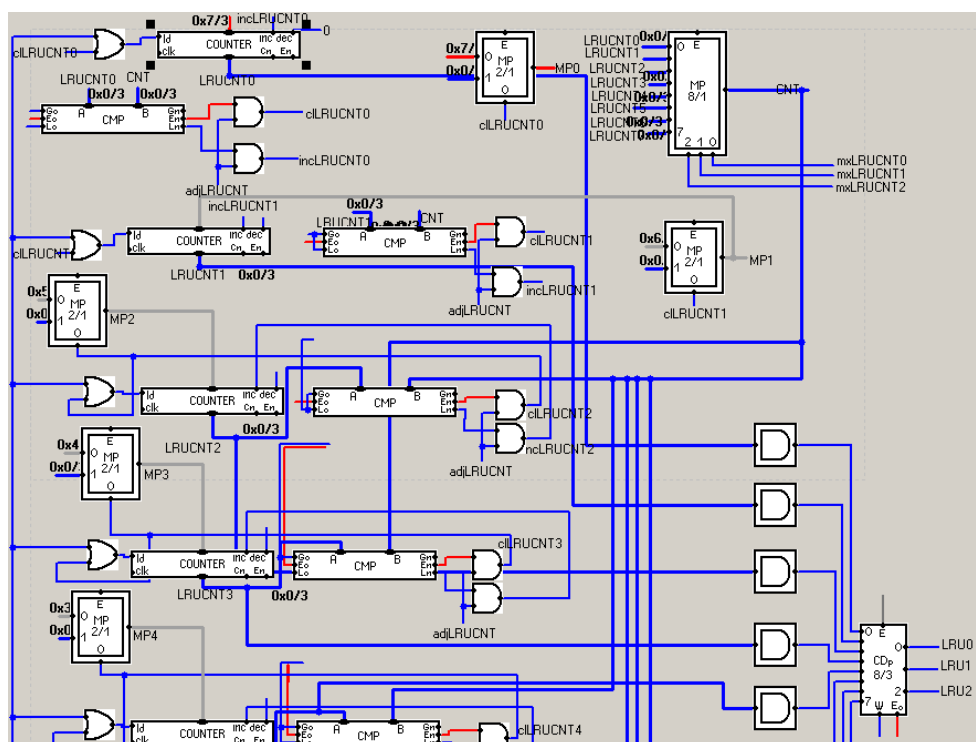


4.2.6 Blok lru kola

Blok je realizovan prema slici iz glave 2.1.3.1 i igra važnu ulogu kod realizovanja algoritma zamene blokova u podmodulu *tag memorija*. Sastoji se od 8 brojača čije sadržaje je potrebno prilikom svakog pristupa keš memoriji ažurirati. Izlazi modula, koriste se za selekciju jednog od ulaza memorije TAG gde se vrši upis ukoliko je to potrebno.

Pošto u standardnoj biblioteci ne postoji konektor za brisanje sadržaja brojača, ovde je problem rešen dodavanjem multipleksera koji na ulaze brojača šalje ili vrednost koju podrazumevano treba upisati ili 0.

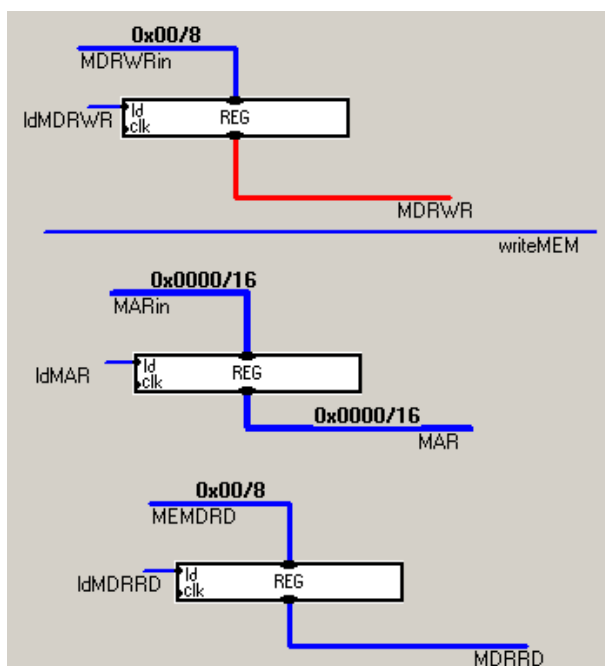
Zbog složenosti šeme, ovde će biti prikazan samo deo.



4.2.7 Blok mem interfejs

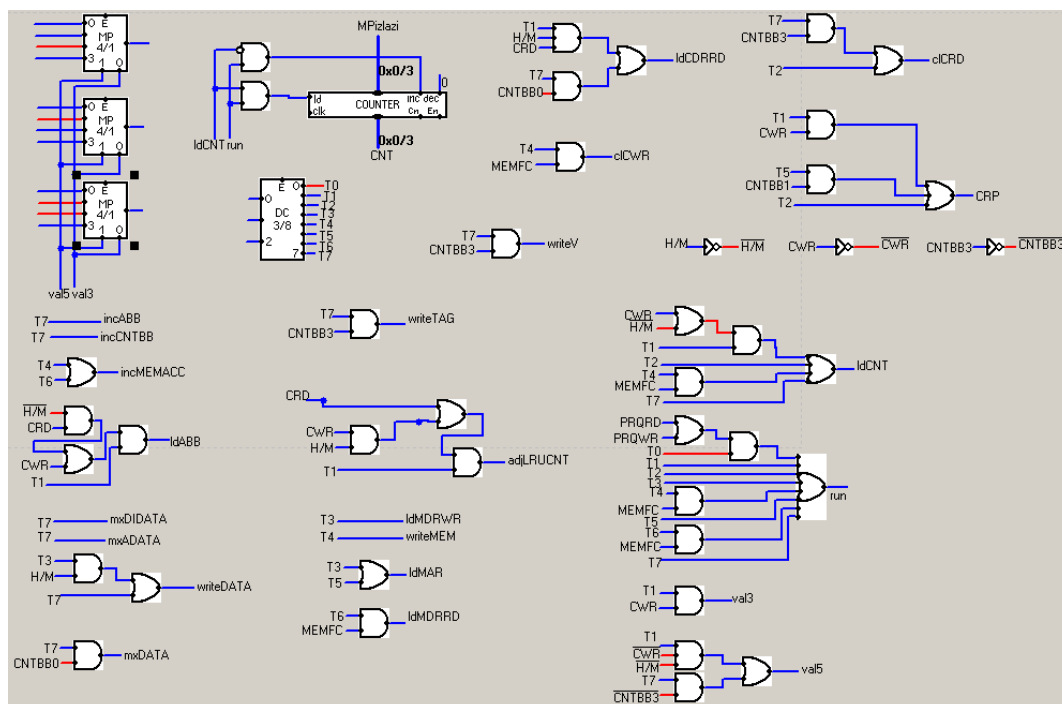
Namena bloka je da omogući komunikaciju modula **KEŠ** i modula **MEM**. Registri koje sadrži ovaj blok koriste se za prenos podataka i to podataka vezanih za adresiranje bajta u memoriji **MEM**, zatim podatka kojeg treba očitati iz modula **MEM** kao i podatka koji treba upisati. Realizacija je izvršena prema slici sa strane 34, iz glave 2.1.3.1.

Simulacija računarskog sistema sa asocijativnom keš memorijom



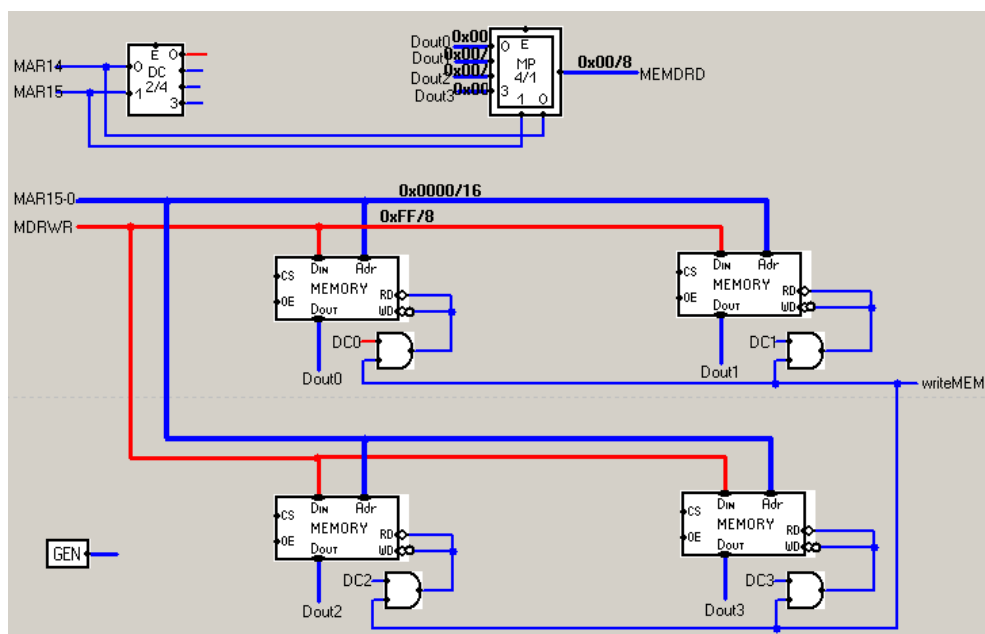
4.2.8 Blok upravljačka jedinica

Blok *upravljačka jedinica* odgovoran je za kreiranje upravljačkih signala kojima se kontroliše rad svi blokova koji čine modul **KEŠ** uključujući i blok *upravljačka jedinica*. **Upravljačka jedinica** može se naći u tri režima rada, režim skoka, režim inkrementiranja i režim mirovanja. Realizacija je izvršena prema slici iz glave 2.1.3.2.



4.3 Memorija MEM

Memorijski modul **MEM** kapaciteta je 64KB i sastoji se od četiri zasebna modula, svaki kapaciteta 16 KB. Adresne linije su širine 16 bita, ali se na liniju adresa vodi samo nižih 14 bita adrese, dok se viša dva bita koriste za adresiranje podmodula u samom modulu **MEM**. Linija za podatke širine je 8 bita. Upravljački signal writeMEM se



korisiti prilikom upisa sadržaja sa linija za podatke MDRWR, adresiranih linijama MAR. Neaktivna vrednost signala writeMEM, je u stvari zadavanje operacije čitanja sa adrese zadate linijama MAR, dok se očitana vrednost pojavljuje na linijama MEMDRD. Modul je realizovan prema slici iz glave 2.1.2.

4.4 Prikaz procesa simulacije keš memorije sa asocijativnim preslikavanjem

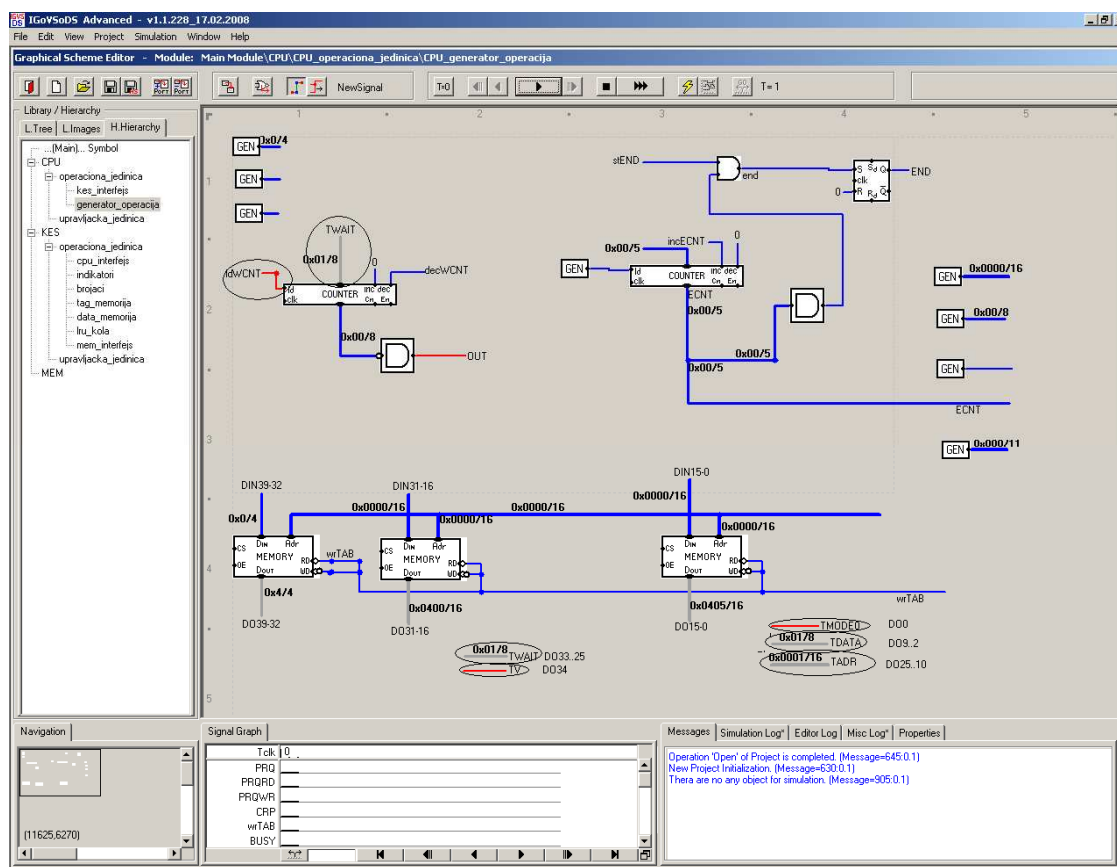
Biće prikazano ponašanje računarskog sistema ukoliko se izvršavaju sledeće operacije:

- upis podatka 01hex na lokaciju 0001hex, uz čekanje 1 Tclk
- čitanje sa lokacije 000Ahex, uz čekanje 2 Tclk
- čitanje sa lokacije 010Ahex, uz čekanje 1 Tclk
- upis podatka 05hex na lokaciju 0009hex, uz čekanje 1 Tclk

Na početku svi ulazi keš memorije su prazni. Dok su popunjeni ulazi memorije TAB kako bi mogle da se generišu operacije koje su gore navedene. Prva četiri ulaza memorije TAB imaju sledeće vrednosti:

4 0400 0405hex, 4 0800 2800hex, 4 0404 2800hex, 4 0400 2415hex

U trenutku T=1 generiše se operacija upisa podatka 01hex na adresu 0001hex, a u brojač WCNT upisuje se 1, jer je potrebno toliko taktova sačekati između završetka jedne i početka nove operacije. Ovo je prikazano na **slici 1**.



Slika 1. Prikaz generisanja operacije upisa

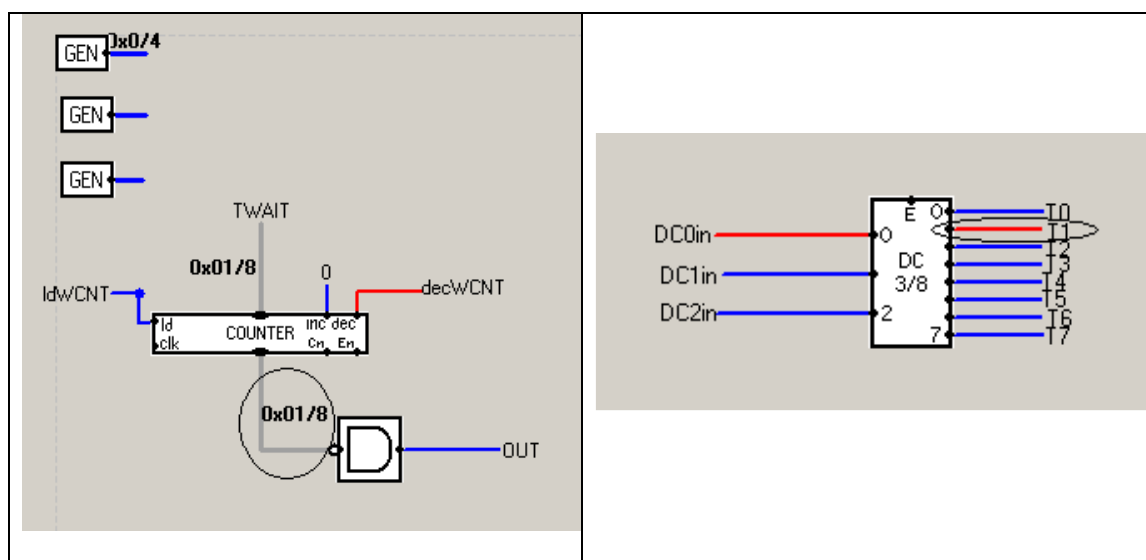
Simulacija računarskog sistema sa asocijativnom keš memorijom

U trenutku T=2 sadržaj je upisan u brojač WCNT, a upravljačka jedinica prelazi u korak 1. Ovo je prikazano na **slici 2**. Zatim se čeka da upravljačka jedinica dođe do koraka 3 kada će generisati signale koji učitalu sadržaje u registre bloka *keš interfejs*.

U taktu T=4 generišu se signali trećeg koraka, a to su signali učitavanja registara bloka *keš interfejs*, ovo je prikazano na **slici 3**.

U taktu T=5 generiše se signal PRQ, a pošto se radi o operaciji čitanja postaje aktivan i signal PRQWR, adresa sa koje se čita nalazi se na linijama TADR registra PAR bloka *keš interfejs*, odnosno na PAR linijama registra CAR bloka *cpu interfejs* (**slika 4**). U ovom taktu vrši se i učitavanje vrednosti brojača bloka *lru kola*, jer je došlo do generisanja prve operacije (**slika 5**).

U trenutku T=6 učitavaju se vrednosti podatka koji se upisuje i adresa na koju se vrši upis u registre PDRWR i CAR, bloka *cpu interfejs*, respektivno, a signal BUSY se postavlja na aktivnu vrednost kako bi signalizirao da će keš memorija biti zauzeta upisom podatka. (**slika 6**).



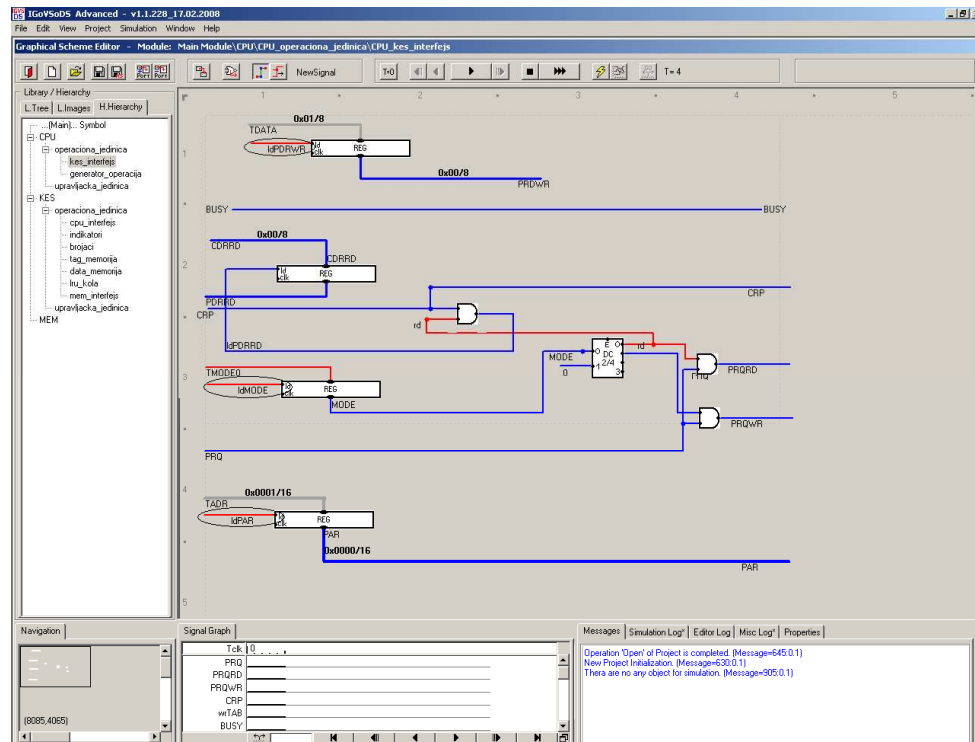
Slika 2 Prikaz upisa vrednosti u brojač WCNT i koraka u kome se nalazi upravljačka jedinica procesora

Signal *ldMDRWR* postavlja se na aktivnu vrednost u trenutku T=7 čime se omogućava upis adrese na kojoj će podatak biti promenjen.

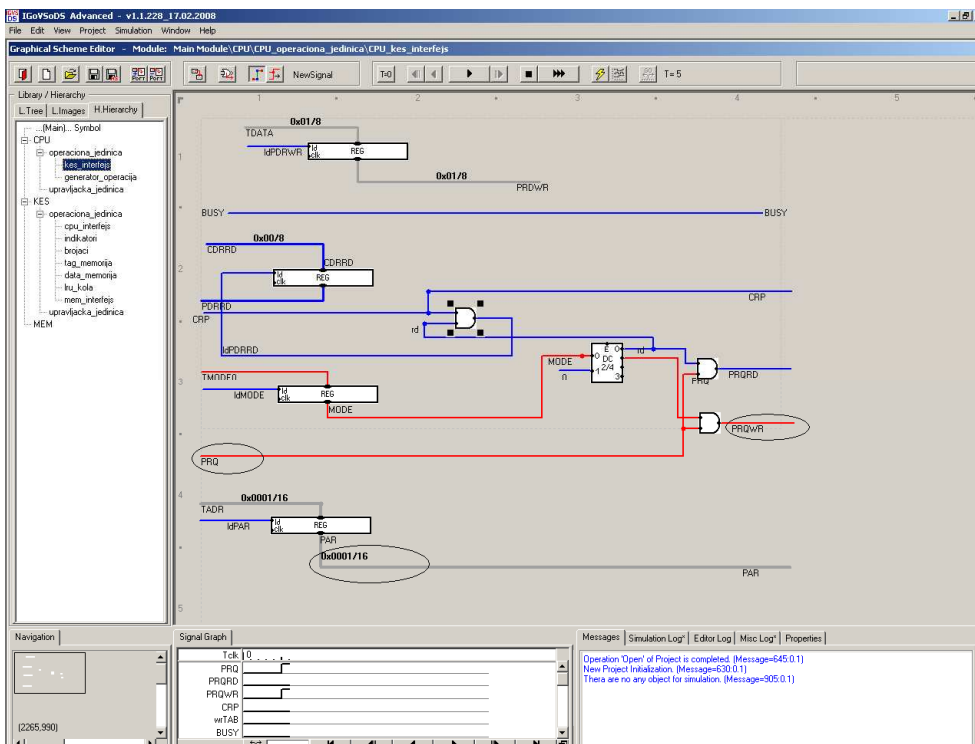
I

U trenutku T=8, generiše se i signal *writeMEM*, nakon koga nastaje upis u memoriju **MEM**, što je prikazano na **slikama 7 i 8**. Zanimljivo je da se ne vrši upis u

Simulacija računarskog sistema sa asocijativnom keš memorijom



Slika 3 Prikaz generisanja signala ldPDRWR, ldMODE, ldPAR



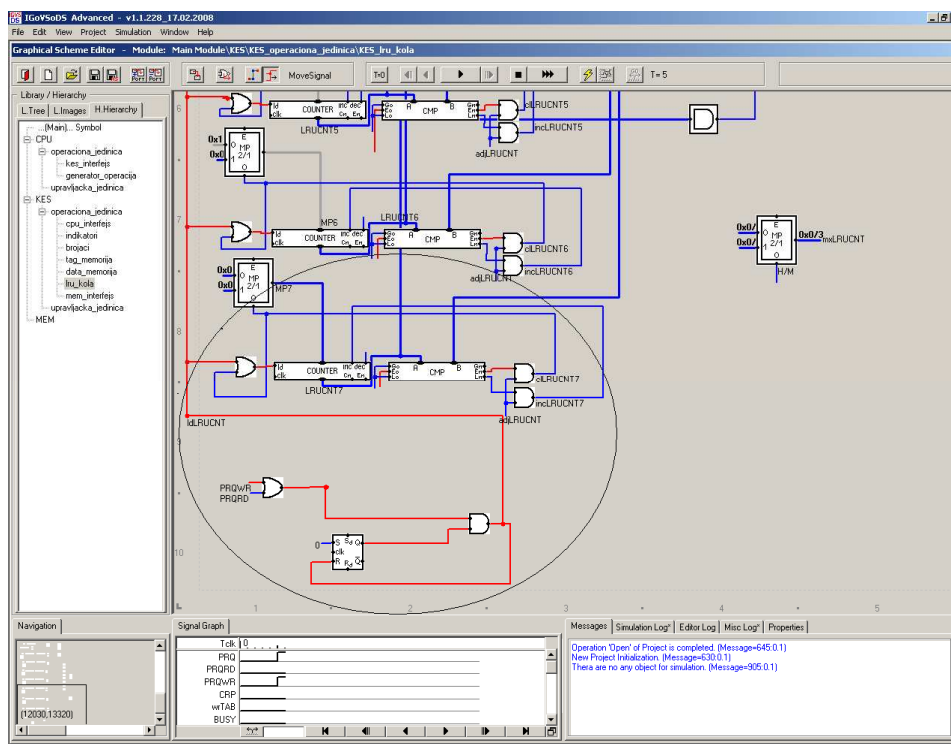
Slika 4 Prikaz sadržaja linija PRQ, PRQWR i PAR

Simulacija računarskog sistema sa asocijativnom keš memorijom

memoriju TAG. Upis traje 10 perioda signala takta koliko se pristupa memoriji **MEM**, za to vreme blok *generator operacija* postavio je novu vrednost na izlaze memorije TAB. Na **slici 9** iz trenutka T=18 vidi se da ni jedan od indikatora ne ukazuje na zauzetost nekog od ulaza memorije TAG, tj svi inikatori su na neaktivnoj vrednosti, što je znak da nije upisano ništa još uvek ni u jedan od ulaza TAG memorije.

Trenutak T=17 je trenutak u kome se upis u memoriju **MEM** završava što uzrokuje generisanje signala MEMFC. Ovo je prikazano na **slici 10**. Na **slici 11** prikazano je trenutno stanje modula **MEM**, sa koje možemo da uočimo da je na adresu 0001hex upisana vrednost 01hex. Nakon ovog trenutka signal BUSY obara se na neaktivnu vrednost i prelazi na novu instrukciju iz bloka generator operacija.

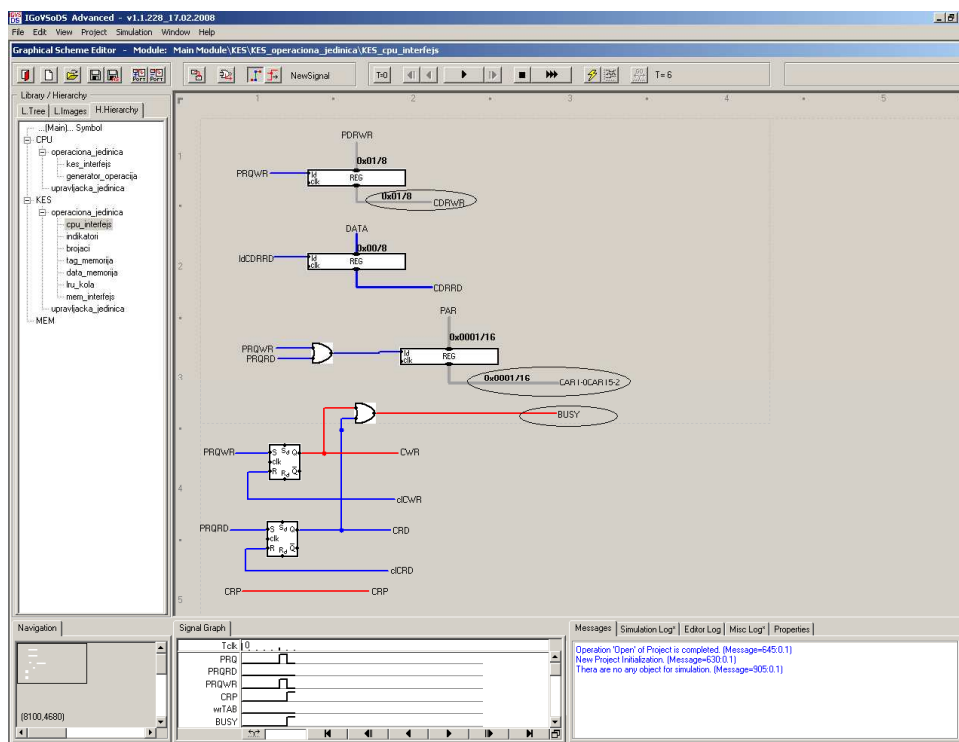
Slika 12 prikazuje generisanja zahteva za čitanje od strane procesora PRQRD. Što odmah uzrokuje učitavanje adresa sa koje se podatak čita u registar CAR bloka *cpu interfejs*. Ovo se dešava u trenutku T=18. U sledećem momentu signal BUSY postaje aktivan, čime se signalizira da keš memorija dovlači podatak iz memorije **MEM**.



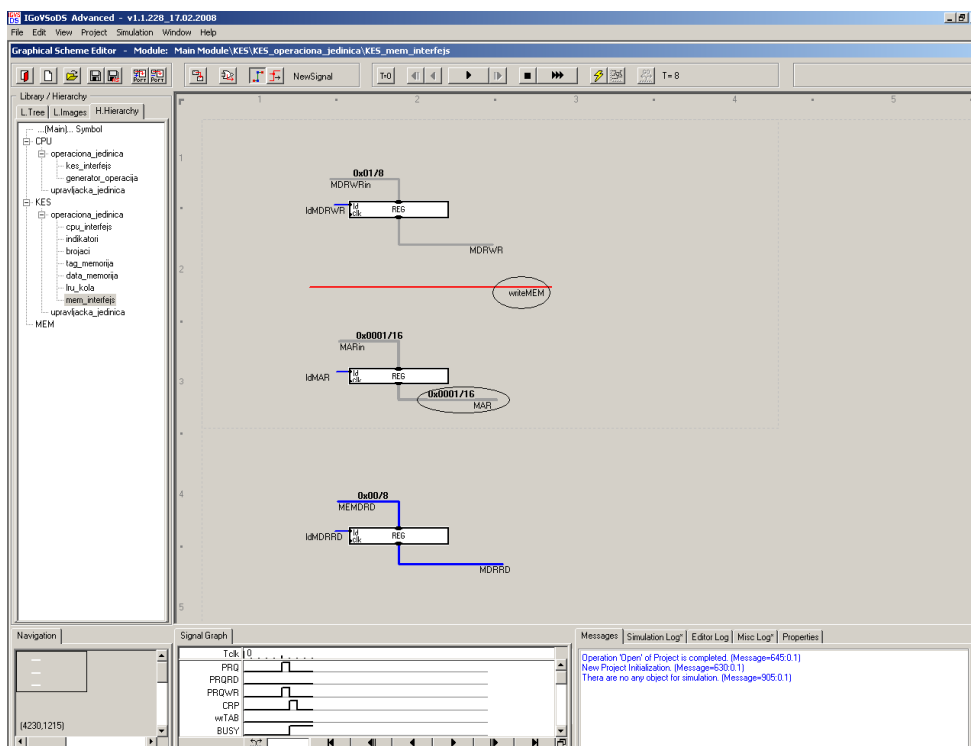
Slika 5 Generisanje signala ldLRUCNT

U trenutku T=21 adrese sa koje se podatak čita nalazi se na linijama MAR registra MAR bloka *mem interfejs*, odnosno linijama MAR15-0 bloka **MEM**. (**slika 13**) Signal writeMEM je neaktivan, pa se zna da se radi o operaciji čitanja. Čitanje počinje od podatka na adresi 000Ahex, što je prikazano na **slici 14** dobijenoj za vreme pristupa memoriji **MEM** u taktu T=30.

Simulacija računarskog sistema sa asocijativnom keš memorijom

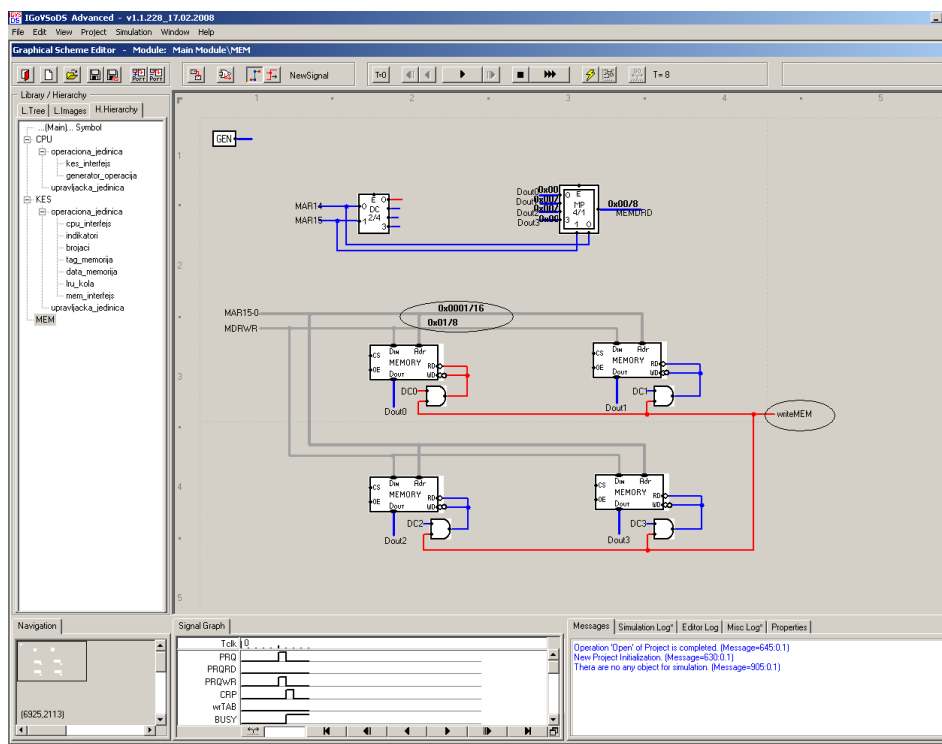


Slika 6 Vrednosti su upisane u registre bloka *cpu interfejs*

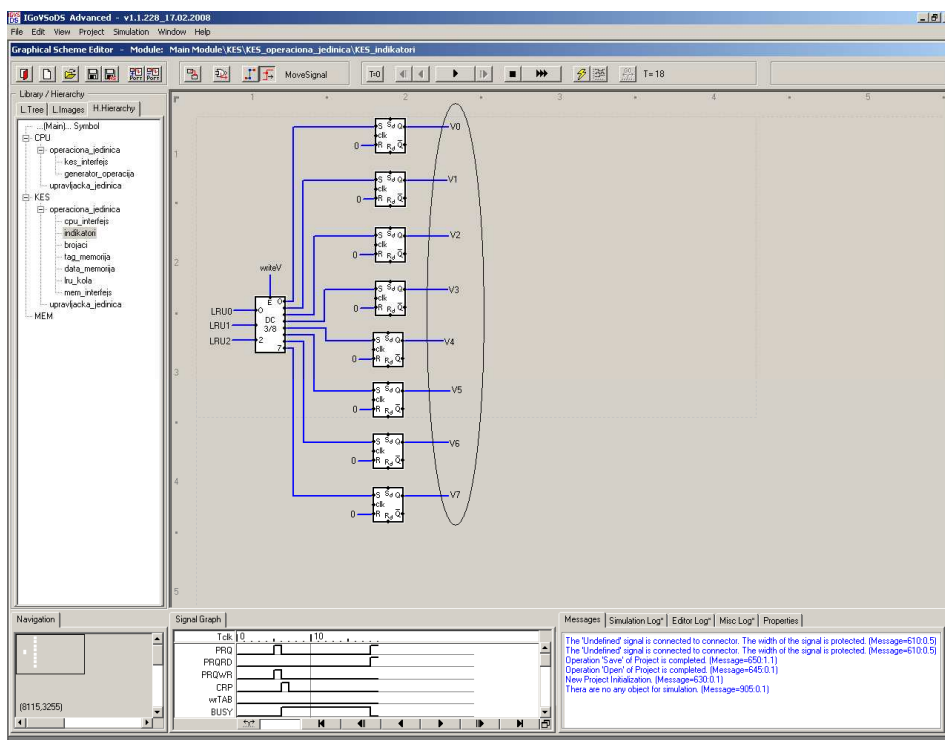


Slika 7 Signal writeMEM je generisan

Simulacija računarskog sistema sa asocijativnom keš memorijom

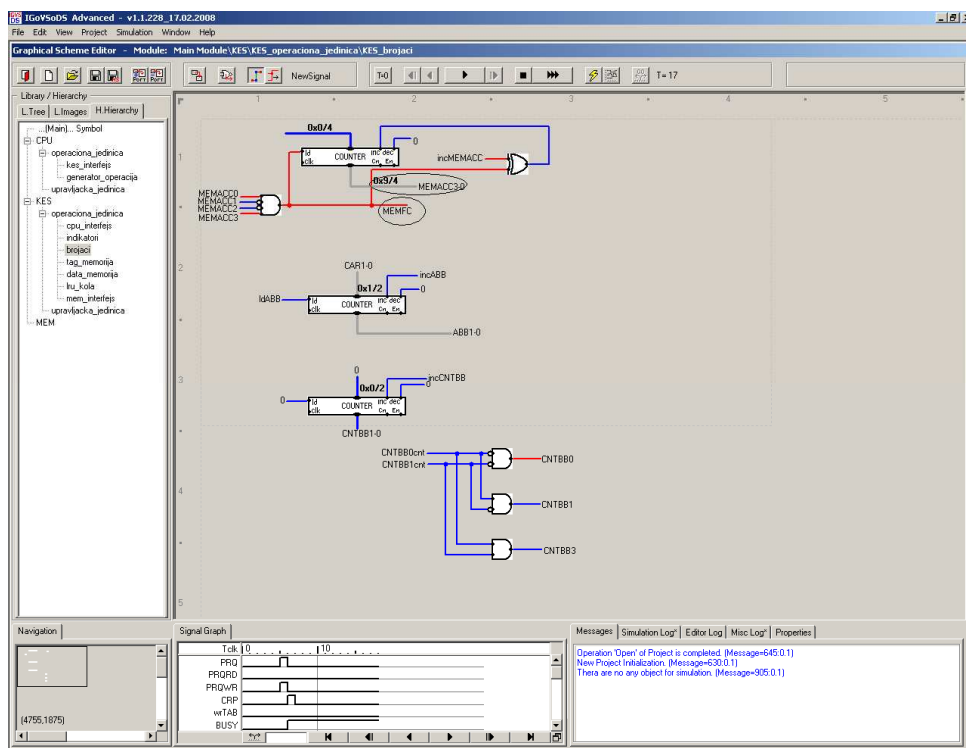


Slika 8 Upis u memoriju *MEM*

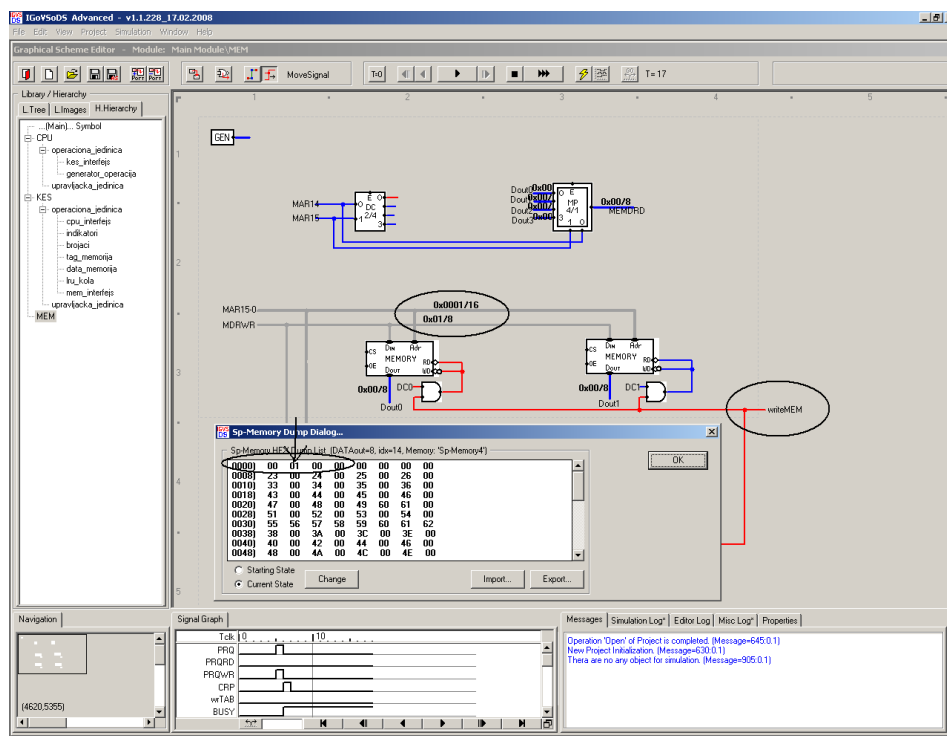


Slika 9 Svi indikatori su na neaktivnoj vrednosti, što znaci da ni jedan ulaz memorije TAG nije validan

Simulacija računarskog sistema sa asocijativnom keš memorijom

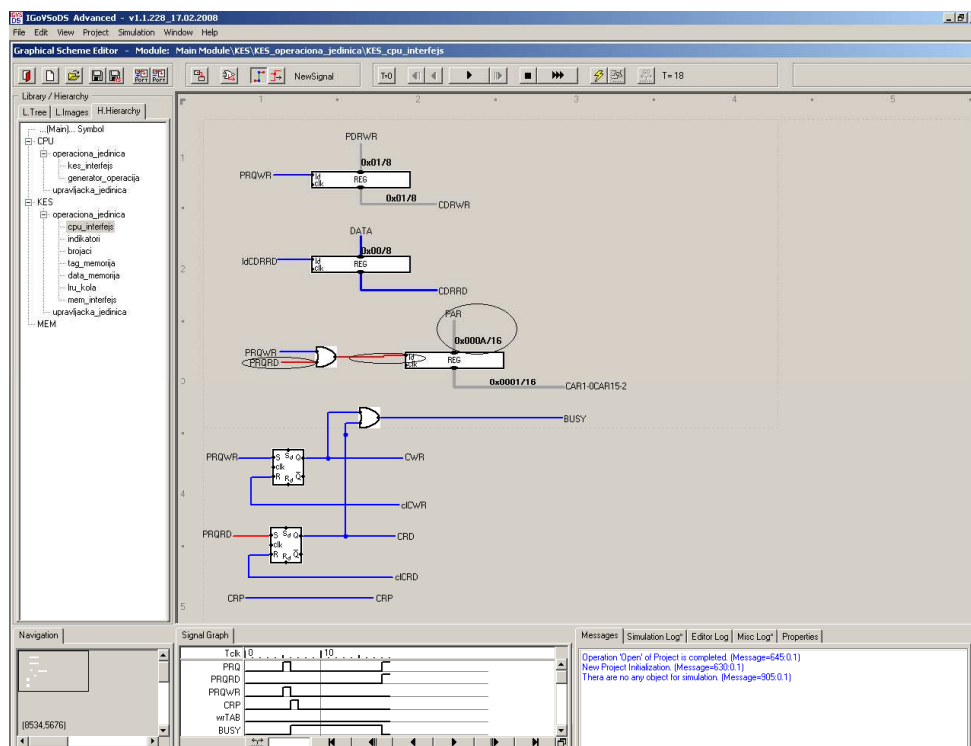


Slika 10 Generisanje signala MEMFC



Slika 11 Sadržaj lokacije 0001hex je 01hex

Simulacija računarskog sistema sa asocijativnom keš memorijom



Slika 12 Generisanje signala učitavanja registra CAR

Slika 15 pokazuje upis sadržaja sa lokacije 000Ahex u modul *data memorija* u trenutku T=31.

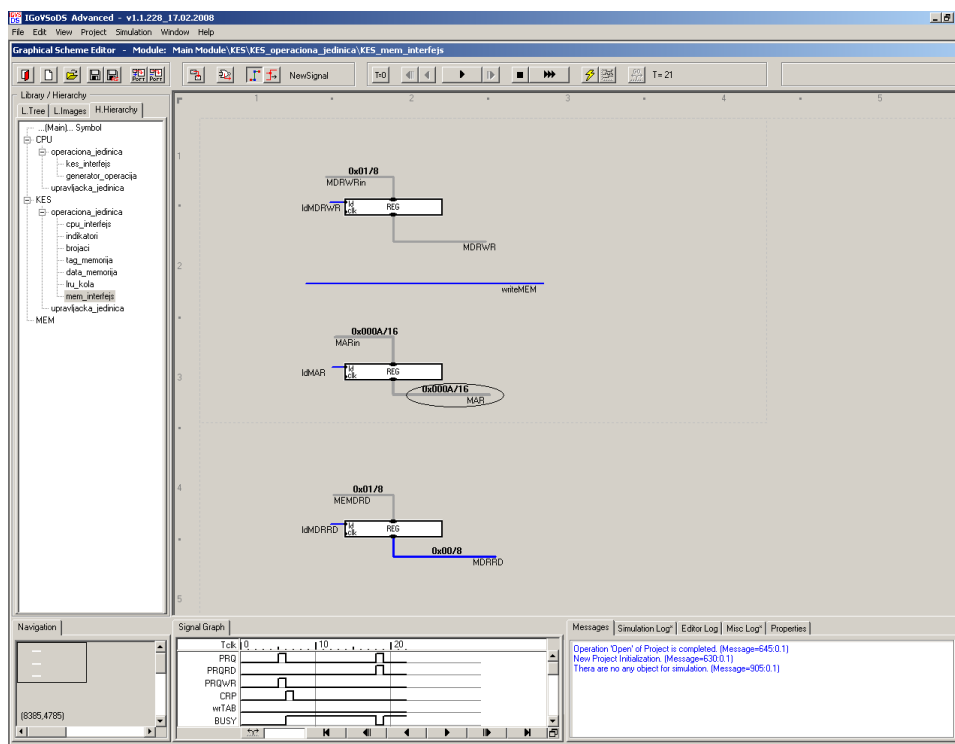
Slika 16 pokazuje upis u modul TAB memorija bloka *generator operacija* očitano podataka preko linija DIN15-0 u trenutku T=33.

Slika 17 prikazuje trenutak T=34 u kome se nastavlja sa dovlačenjem preostalih bajtova bloka.

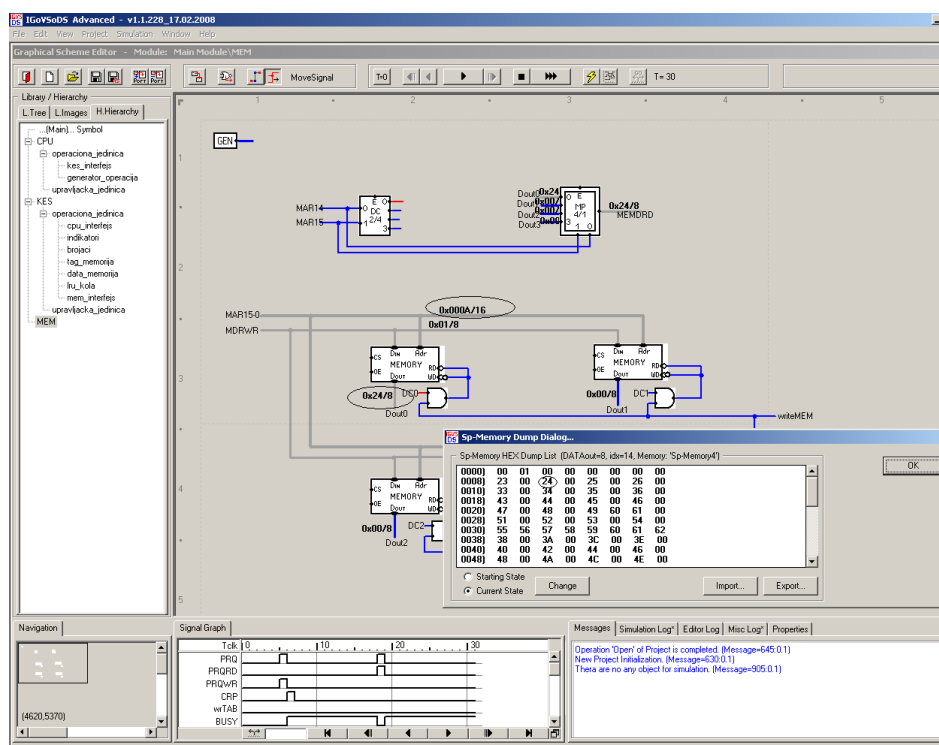
Nakon dovlačenja i sadržaja sa lokacije 0008, dovlači se i poslednji bajt bloka 2hex, a to je sadržaj sa lokacije 9hex. Na **slici 18** možemo videti i da je brojač CNTBB odbrojao do 3, što znači da se prenosi i četvrti bajt podataka. T=59.

Slika 19 završeno je sa dovlačenjem svih bajtova bloka, pa se generiše signal writeTAG u taktu T=67, te se u ulaz jedan upisuje vrednost sa linija DI, a to je 0002 hex.

Simulacija računarskog sistema sa asocijativnom keš memorijom

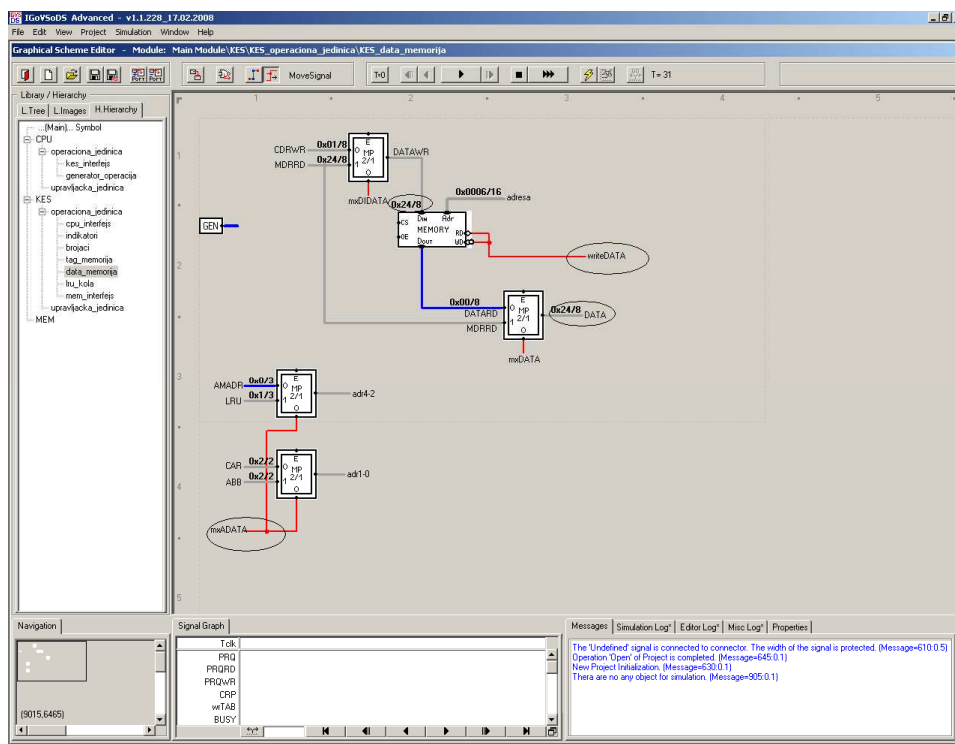


Slika 13 Na izlaznim linija registra MAR je vrednost 000Ahex

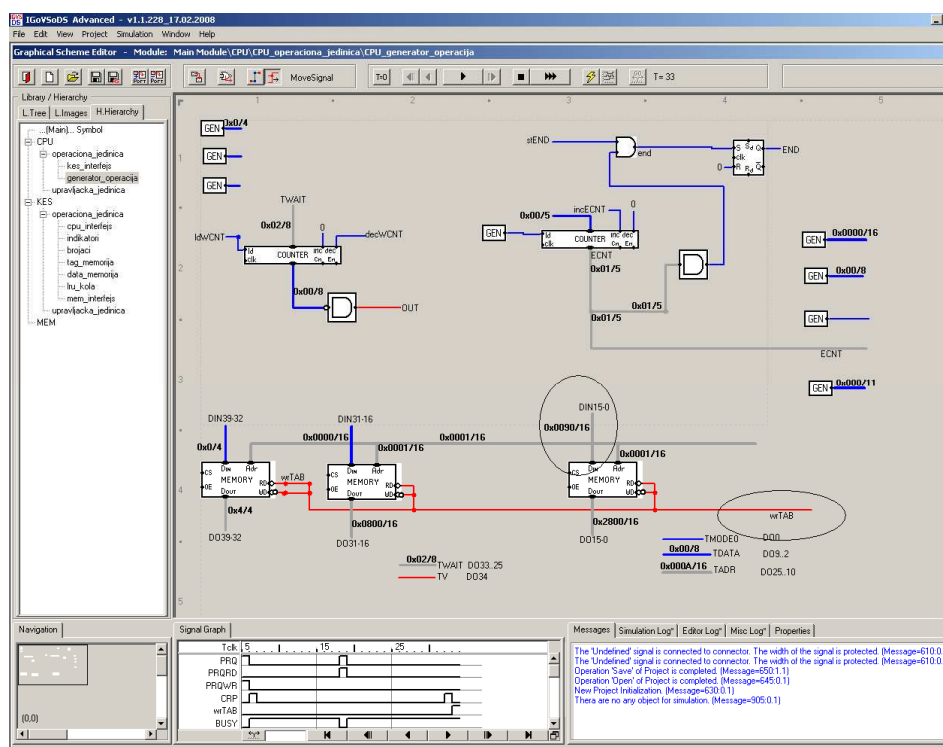


Slika 14 Na linijama Dout nalazi se vrednost 24hex, čto je sadržaj lokacije 000Ahex

Simulacija računarskog sistema sa asocijativnom keš memorijom

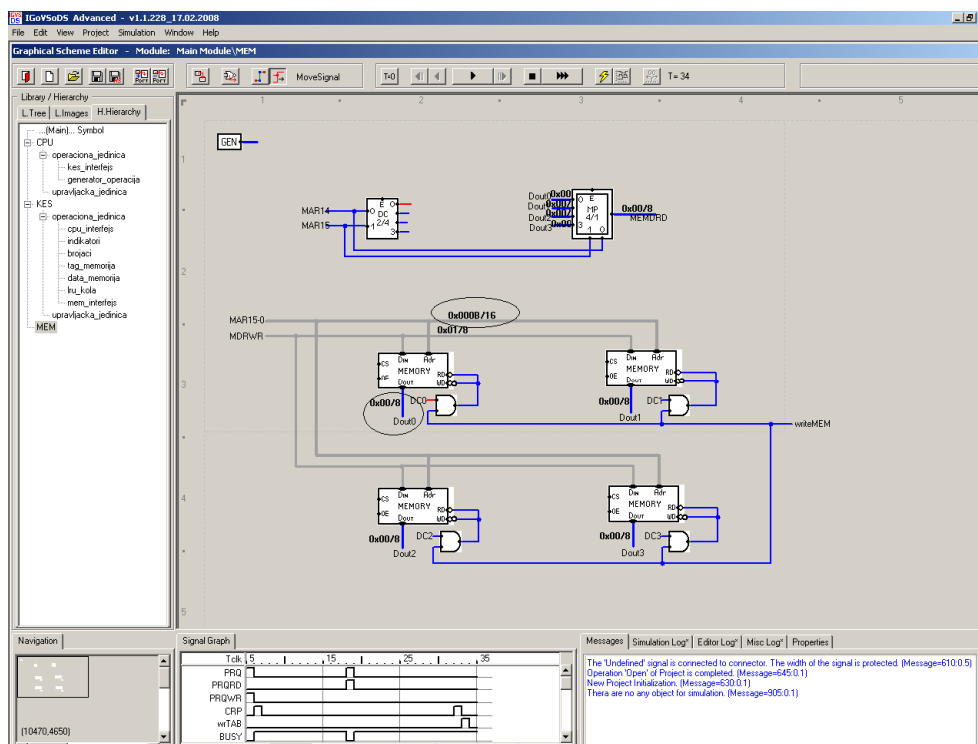


Slika 15 u modula DATA memorija upisuje se vrednost 24hex

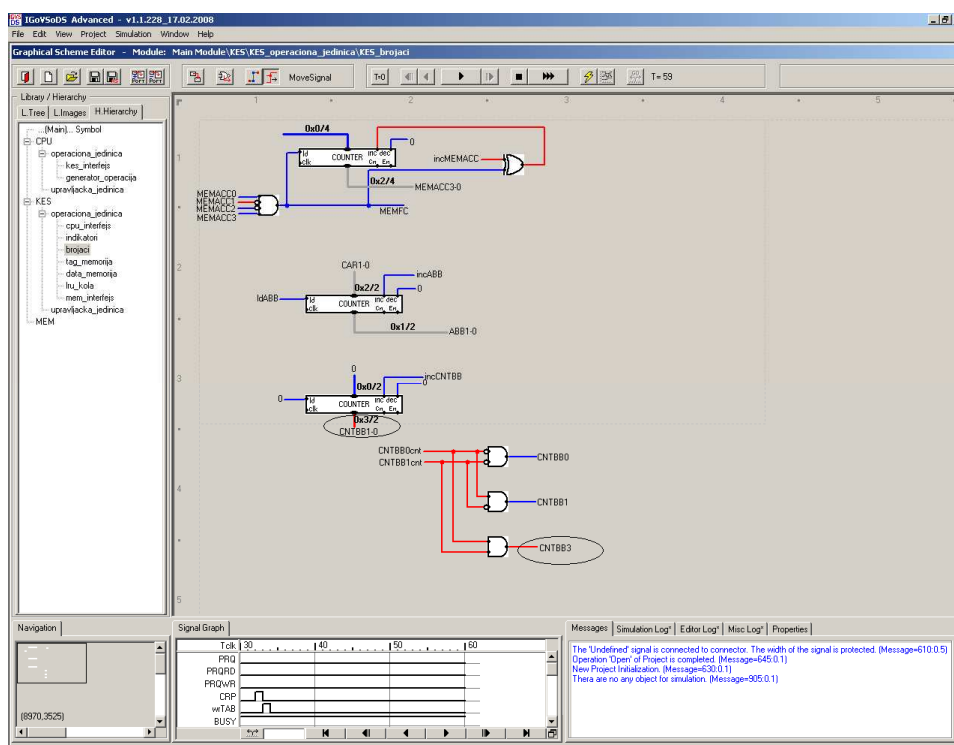


Slika 16 Upis vrednosti 24hex na adresu 0001hex memorije TAB

Simulacija računarskog sistema sa asocijativnom keš memorijom

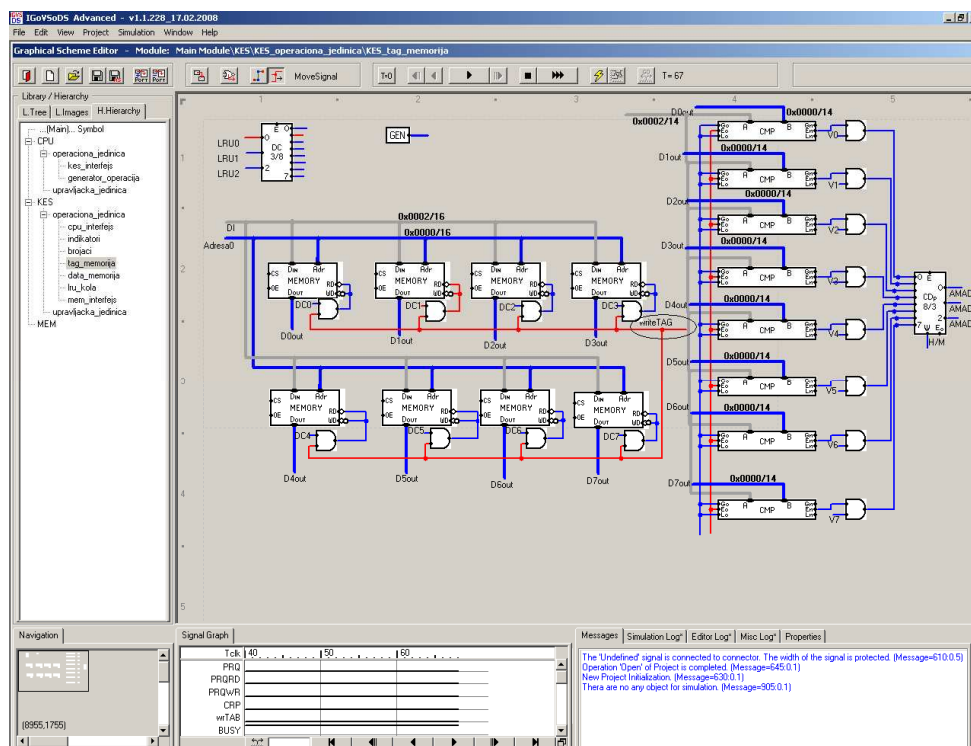


Slika 17 Dovlačenje bajta sa lokacije 000Bhex



Slika 18 Brojač CNTBB je odbrojao do 3 što znači da se dovlači četvrti bajt bloka 2

Simulacija računarskog sistema sa asocijativnom keš memorijom



Slika 19 Generisan je signal writeTAG

Takt $T=68$, u modul *tag memorija* upisan je broj bloka čiji se sadržaji nalaze u modulu data memorija. Blok sa brojem 0002hex se nalazi u prvom ulazu keš memorije, što znači da su podaci u data memoriji čuvaju na lokacijama počev od adrese 4hex. Ovo vidimo na **slikama 20 i 21**.

Na **slici 22** prikazan je takođe trenutak $T=68$, ali blok indikator, gde možemo da vidimo da je prvi ulaz keš memorije sada validan.

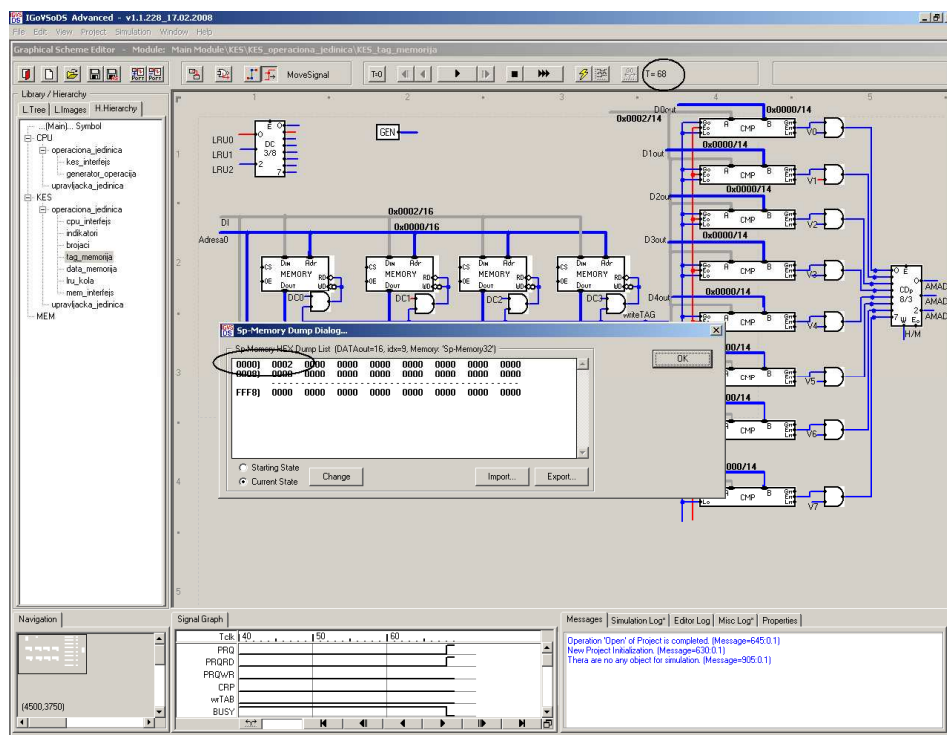
U trenutku $T=69$, opet je generisana operacija čitanja. Kako nije došlo do saglasnosti sa nekim od ulaza memorije TAG, vrši se ažuriranje brojača bloka *lru kola*.

Na **slici 23** prikazano je kako signal adjLRUCNT utiče na vrednosti brojača. Ovaj signal izaziva resetovanje, brisanje sadržaja samo jednog brojača dok inkrementira sadržaj svih ostalih.

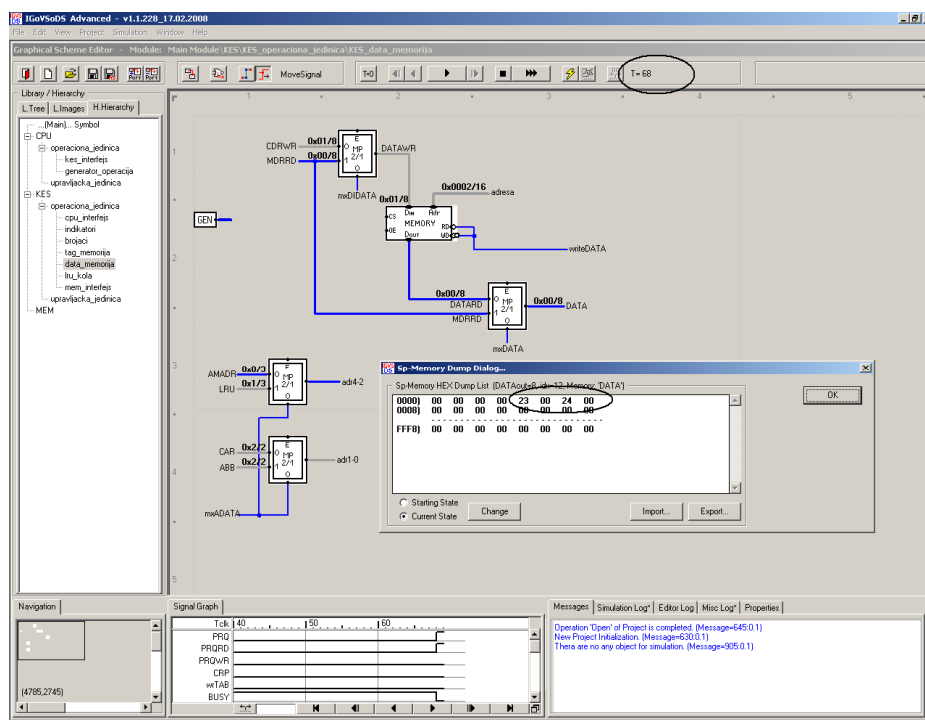
Po ažuriranju sadržaja brojača, pristupa se dovlačenju traženog bajta u modul *data memorija*, a zatim se i vrši upis u memoriju TAB u trenutku $T=83$. (**slika 24**)

Posle trenutka $T=83$, pristupa se dovlačenju preostalih bajtova bloka 42hex. Ovi bajtovi smeštaju se u prvi slobodan ulaz memorije DATA određen sadržajem na linijama LRU. U ovom slučaju to je ulaz dva. Na **slikama 25 i 26** prikazani su sadržaji drugog ulaza memorije TAG i DATA respektivno u vremenskom trenutku $T=118$.

Simulacija računarskog sistema sa asocijativnom keš memorijom

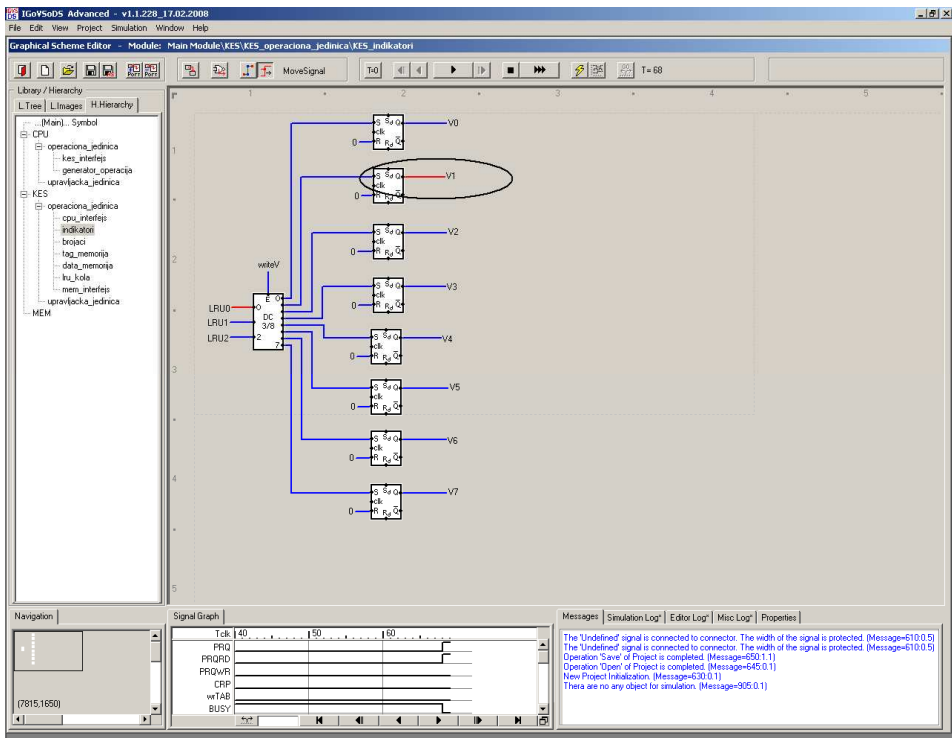


Slika 20 Sadržaj prvog ulaza tag memorije je 0002hex

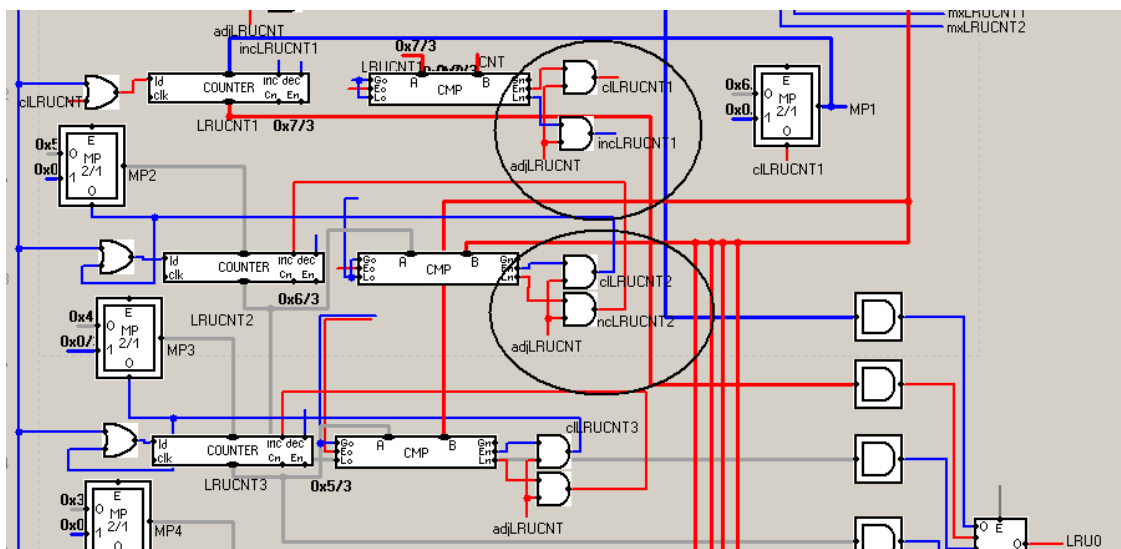


Slika 21 Sadržaj prvog ulaza memorije DATA je sadržaj drugog bloka memorije **MEM**

Simulacija računarskog sistema sa asocijativnom keš memorijom

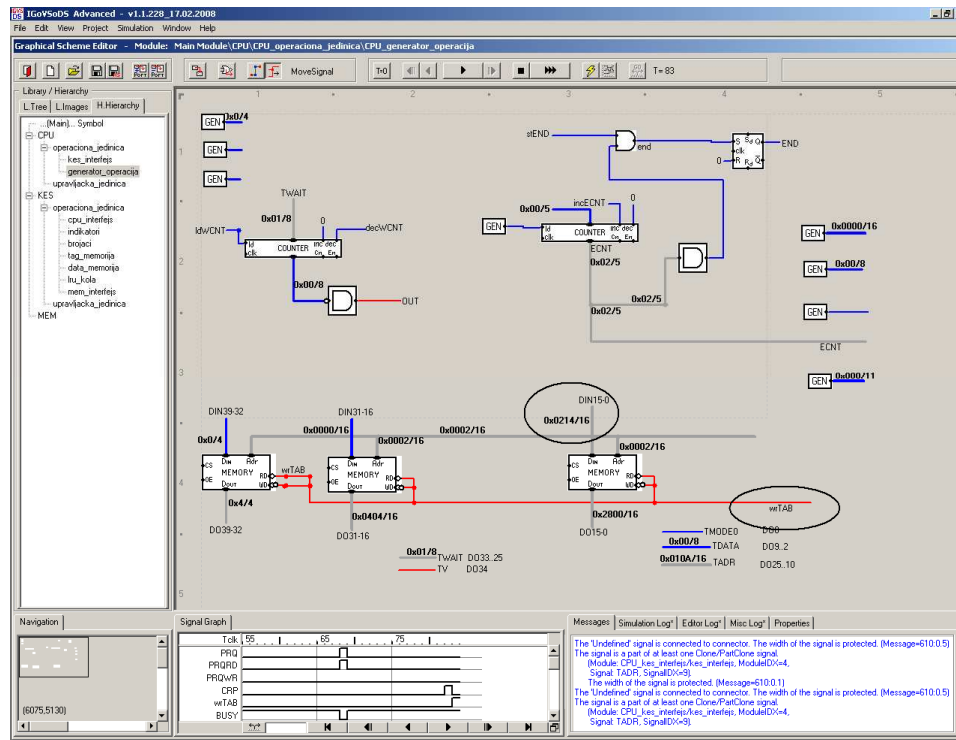


Slika 22 V1 signal je na aktivnoj vrednosti, došlo je do upisa u TAG i DATA memoriju.

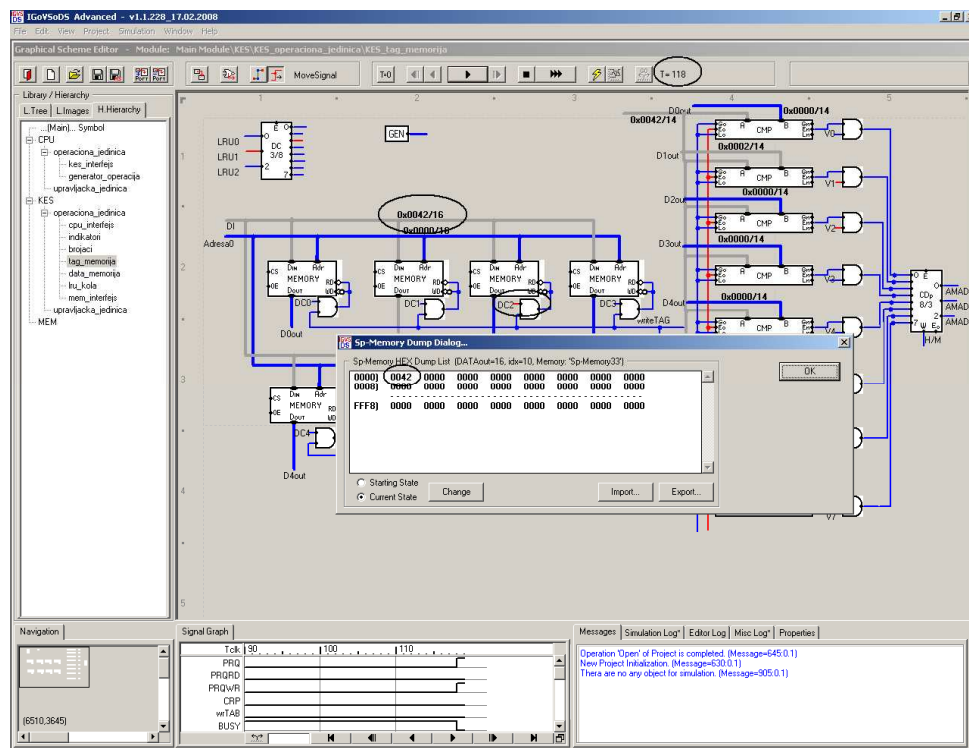


Slika 23 signal adjLRUCNT može izazvati brisanje sadržaja jednog brojača ili inkrementiranje sadržaja svih ostalih

Simulacija računarskog sistema sa asocijativnom keš memorijom

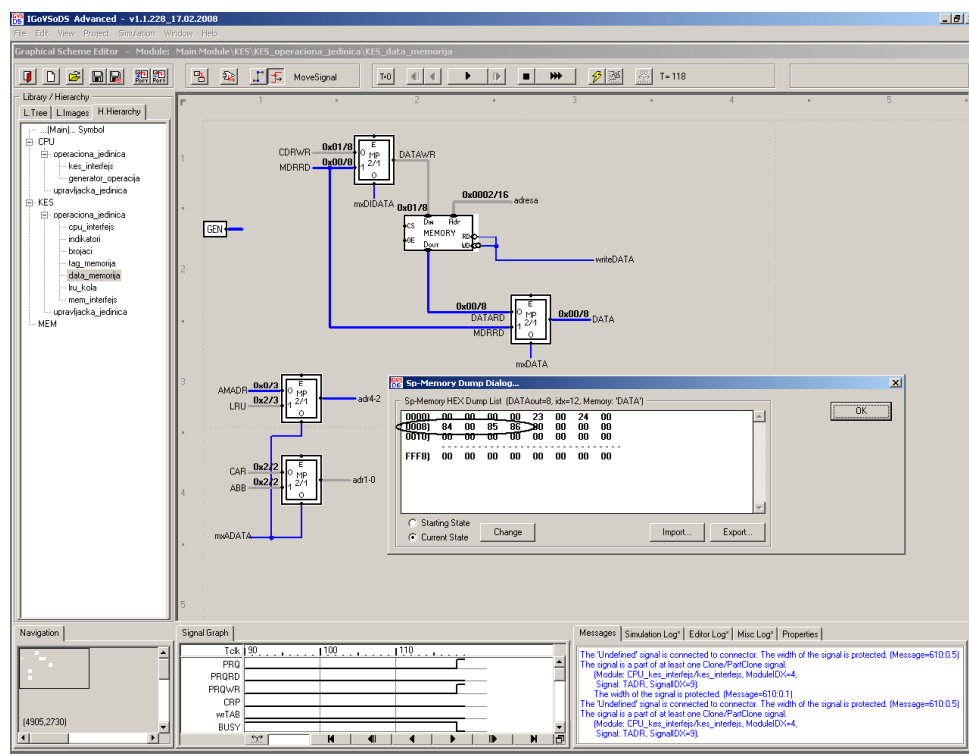


Slika 24 Upis vrednosti 85hex u memoriju TAB



Slika 25 Sadržaj drugog ulaza memorije TAG je 0042hex

Simulacija računarskog sistema sa asocijativnom keš memorijom



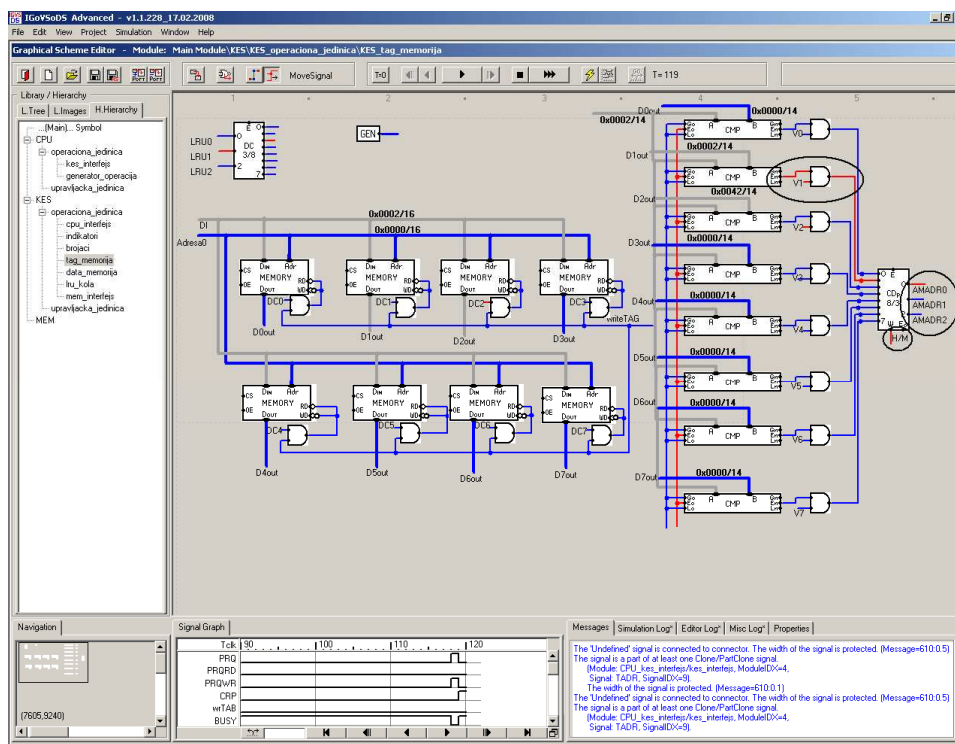
Slika 26 Sadržaj drugog ulaza memorije DATA je sadržaj 42hex bloka memorije MEM

U trenutku $T=119$, nakon generisanja nove operacije, operacije upisa i postavljanja signala **BUSY** na aktivnu vrednost, na linijama **DI** bloka *tag memorija* našla se vrednost 0002hex. Pošto je ova vrednost već upisana u prvi ulaz memorije **TAG**, generiše se signal podudaranja M_1 , a pošto je ovaj ulaz tag memorije validan i V_1 je na aktivnoj vrednosti, zbog toga se linijama **AMADR** nalazi vrednost 1. Na **slici 27**, jasno se vide ove promene.

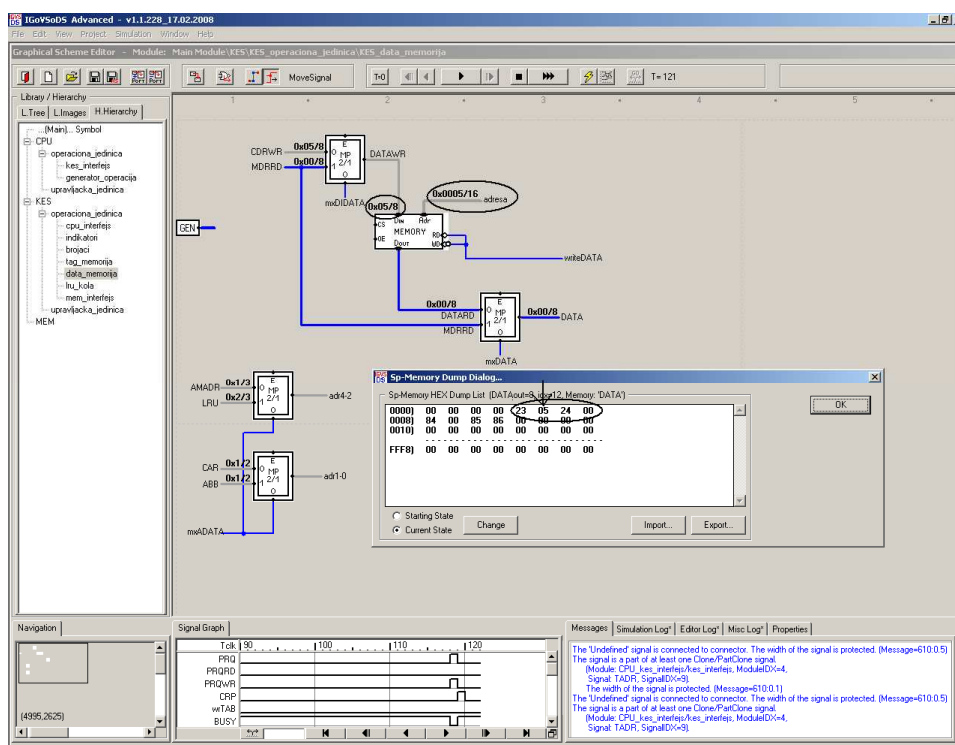
Slika 28 prikazuje da je u trenutku $T=121$ završen upis u memoriju **DATA** vrednosti 0005hex na lokaciju u ulazu 1 koja predstavlja sadržaj bloka 2. Adresa u okviru samog modula *data memorija* je 0005hex.

Posle isteka vremena potrebnog za pristup memoriji **MEM**, podatak je upisan i u samu memoriju **MEM**, jer se primenjuje pristup *upiši-skroz*. U trenutku $T=132$ izmenjen je sadržaj memorije **MEM** što se vidi na **slici 29**.

Simulacija računarskog sistema sa asocijativnom keš memorijom

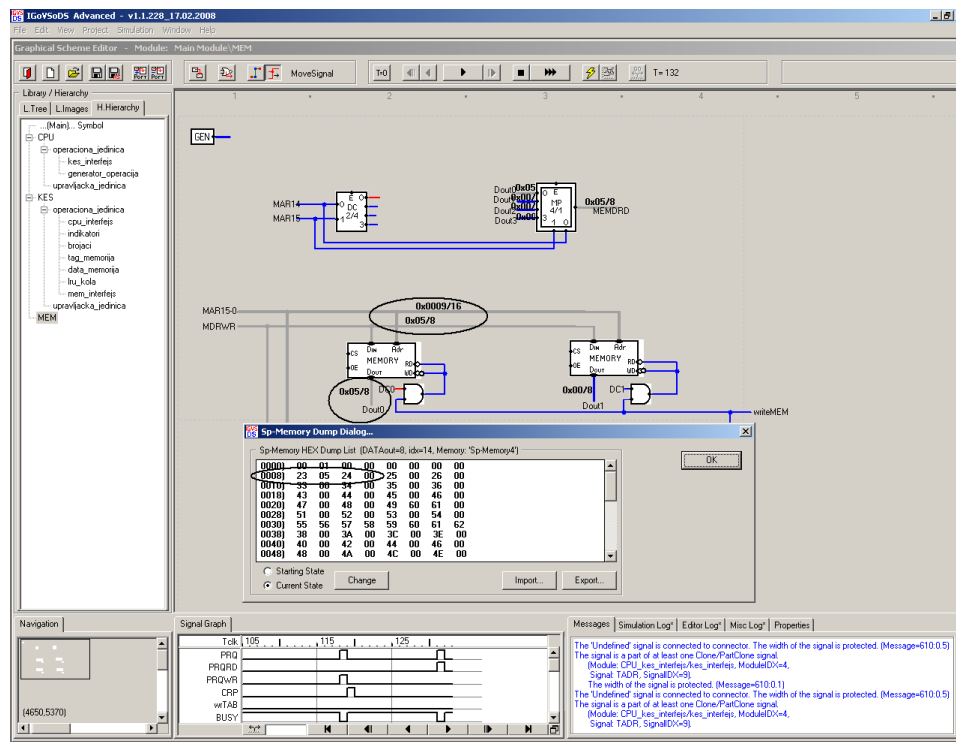


Slika 27 Otkrivanje saglasnosti u prvom ulazu bloka tag memorija



Slika 28 Upis vrednosti u blok data memorija je završen

Simulacija računarskog sistema sa asocijativnom keš memorijom



Slika 29 Sadržaj bloka 2 je promenjen

5 Zaključak

Ne sumnjivo je da je rad procesora usko povezan sa operativnom memorijom, sem vremena koje procesor utroši na izvršavanje samih operacija nad podacima, na njegovo ukupno vreme rada utiče i sama „komunikacija“ sa memorijom, tačnije čekanje na potrebne podatke da budu prebačeni iz operativne memorije i predate procesoru na korišćenje. S obzirom da pristup memoriji može da traje po nekoliko perioda signala takta, udeo vremena koje procesor provede „besposlen“ čekajući na podatak može da bude značajan. U zavisnosti o kakvom se formatu instrukcija radi, za jednu instrukciju ponekad je potrebno više puta pristupati memoriji.

Razvojem tehnologija, razlika u brzinama rada procesora i memorije postajala je sve veća i sve je veći bio problem svesti odnos broja pristiglih zahteva za podatke i broja opsluženih zahteva od strane memorije na što minimalniji. Kao odgovor na problem osmišljena je *keš memorija*.

Keš memorija predstavlja posrednika u komunikaciji procesora i memorije. Njena glavna uloga je da olakša i ubrza rad procesora. Najčešće je organizovana hijerarhijski u dva nivoa rada i sastoji se od više manjih memorija. Jedna od najzanimljivijih osobina keš memorije je da je vreme pristupa keš memoriji mnogo kraće od vremena pristupa standardnoj memoriji. Ovo vreme pristupa smanjivalo se sa razvojem keš memorija, tako da današnje keš memorije rade na brzinama nešto manjim ili istim na kojima radi procesor. Baš zbog kraćeg vremena pristupa, cena keš memorije po bitu je znatno viša od cene operativne memorije, ali su keš memorije manjeg kapaciteta.

Keš memorija sadrži određene blokove memorije, pri generisanju adrese od strane procesora, najpre se vrši provera da li se traženi podatak nalazi u keš memoriji. Ako se podatak pronađe, dalje se obavlja komunikacija samo između keš memoriji i procesora, ako se dogodi suprotno, traženi podatak dovlači se u keš memoriju, kako bi se i u ovom slučaju komunikacija opet svela na razmenjivanje signal procesora i keš memorije. Kod modernih računara čak 80% vremena pristupa memoriji je vreme pristupa keš memoriji, što je odlika stepena usavršenosti savremenih keš memorija i mogućnosti da svedu *miss rate* na što manji broj.

Kod same realizacije keš memorije poželjno je voditi računa o tipu podataka sa kojima procesor radi, kao i sa osobinama koje oni ispoljavaju. Podaci mogu ispoljavati prostorni lokalitet, što predstavlja izdavanje zahteva za podacima sa sukcesivnih adresa ili vremenski lokalitet, što podrazumeva ponavljanje zahteva za istim podacima u određenom vremenskom intervalu. Sagledavajući osobine sistema, potrebno je naći najbolje rešenje koje će u što većoj meri poboljšati performanse. U sistemima gde se mogu sresti obe vrste lokaliteta keš memorija podeljena je na dve keš memorije, koje su prolagođene podacima i raličito su realizovane, te se u jednu keš memoriju smeštaju posadaci sa prostornim lokalitetom, a u drugu sa vremenskim lokalitetom.

Slojevita struktura keš memorija utiče na brzinu. Tako se keš memorija deli na podkeševe, koji su podeljeni u nivoe ili slojeve, sa porastom nivoa smanjuje se brzina

Simulacija računarskog sistema sa asocijativnom keš memorijom

keša, ali raste kapacitet. Podaci se prvo traže u najbržoj memoriji, a to je keš memorija prvog nivoa, zatim u ostalim nivoima i na kraju u operativnoj memoriji. Ponekad se pokazuje kao dobro rešenje i da se isti blokovi nalaze u različitim nivoima, ali to je sve stvar procene i izbora načina realizacije. Ovakvom organizacijom keš memorija pokušava se da se nađe najbolji kompromis između cene, brzine i kapaciteta.

Postavlja se pitanje kako rešiti glavni problem keš memorija, a to su konflikti i *miss*-evi? Da bi se smanjili, potrebno je koristiti keš memorije što je moguće većeg kapaciteta i veće asocijativnosti. Mana keš memorija sa boljim navedenim karakteristikama je što cena raste sa povećanjem pogodnosti koje keš memorija nudi. Drugi način da poboljšamo performanse, jeste dobro poznavanje same realizacije keš memorije i optimizacija koda. Ovo podrazumeva raspoređivanje koda kako bi se on što bolje prilagodio integrisanoj keš memoriji u sistem koji koristi kod. Ovo je možda jedan od najjeftinijih načina za postizanje ubrzanja izvršavanja instrukcija, ali zahteva dobro poznavanje ne samo softvera nego i hardvera od strane programera.

U ovom radu predstavljena je keš memorija sa asocijativnim preslikavanjem koja sa jedne strane može da ima dobar *hit rate*, ali istovremeno, ona je najsloženija za realizaciju, a zbog upotrebe asocijativnih memorijskih modula je relativno skupa. Za simulaciju celog sistema korišćen je alat IGoVSoEDS koji omogućava razumevanje rada ove keš memorije, lako sagledavanje rezultata simulacije, ali i jednostavan pregled same arhitekture sistema. Interfejs alata prilagođen je korisniku, a sve funkcije su pristupačne i intuitivne.

Buduće keš memorije svakako će imati za cilj postizanje dobrih performansi računaskog sistema i neutralizaciju razlike u brzinama procesora i memorije. Svakako na razvoj imaće uticaj i potrebe korisnika u pravac u kom će teći razvoj računara. Smatra se da će zbog tendencije razvoja procesora sa više jezgara u budućnosti javiti potreba za više nivoa keševa. Takođe, zahtevaće se keš memorije što većeg kapaciteta i sa što većom asocijativnošću. Danas se u normalnoj upotrebi mogu naći računari i sa tri nivoa keša.

6 Literatura

1. J.Đorđević, *Arhitektura i Organizacija Računara*, 2005-2006.
2. J.Đorđević, *Opis računarskih sistema sa keš memorijom*, 2006
3. Nenad Grbanović, *Interaktivni generator vizuelnih simulatora digitalnih sistema (IGoVSoDS)*, doktorska disertacija u izradi, Elektrotehnički fakultet, Beograd, 2007.
4. Boško Nikolić, Jovan Đorđević, Nenad Grbanović, *Vizuelni simulatori u nastavi arhitekture i organizacije računara*, Zbornik radova ETRAN 2006, Beograd, Srbija, 2006.
5. Nenad Grbanović, Boško Nikolić, Jovan Đorđević, *Vizuelni simulatori digitalnih modula*, Zbornik radova ETRAN 2006, Beograd, Srbija, 2006.
6. *CPU Cache*
http://en.wikipedia.org/wiki/CPU_cache
7. *System Cache*
<http://www.pcguide.com/ref/mbsys/cache/index.htm>
8. *Instruction Cache Memory Issues in Real-Time Systems*, licthesis.ppt
9. *Cache Hierarchy Features*, L3_Cache_Memory.ppt
10. "Cache Behavior of Network Protocols"
<http://www.research.ibm.com/people/n/nahum/publications/sigmetrics97-cache.pdf>