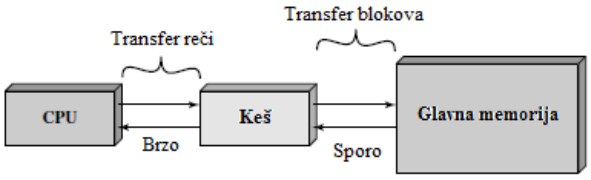
8.

KEŠ MEMORIJA

Namena keš memorije je da pruži brzinu koja je blizu onoj koju imaju najbrže raspoložive memorije, a u isto vreme da obezbedi veliki kapacitet memorije po ceni jeftinih vrsta poluprovodničkih memorija.

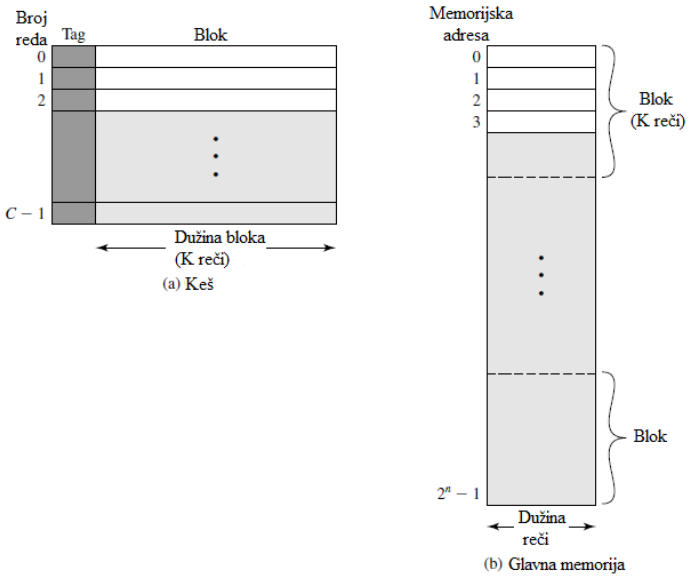
Keš memorija sadrži kopiju delova glavne memorije. Kada procesor pokuša da čita reči iz memorije, prvo se proverava da li je ta reč u kešu, i ako jeste, reč se isporučuje procesoru. Ako nije, blok glavne memorije, koji se sastoji od nekog fiksnog broja reči, se učitava u keš i onda se reč isporučuje procesoru (slika 1). Zbog fenomena lokalnosti reference, kada se blok podataka donese u keš da bi se zadovoljila jedna referenca memorije, verovatno će biti budućih referenci na tu istu memorijsku lokaciju ili na drugu reč u bloku.



Slika 1. Odnos procesora, keša i glavne memorije

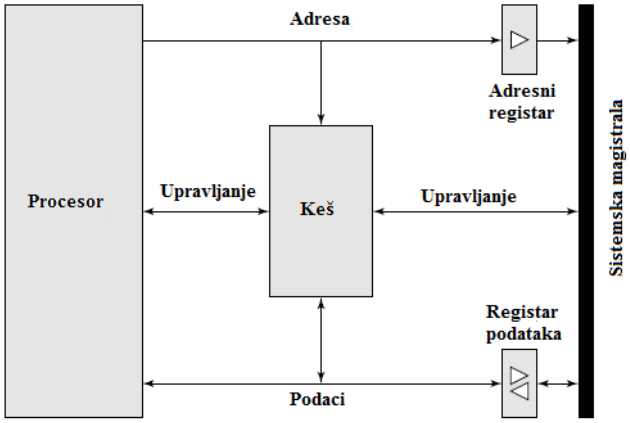
Glavna memorija se sastoji od 2^n adresibilnih reči, gde svaka lokacija (reč) ima jedinstvenu n-bitnu adresu (slika 2). U svrhu preslikavanja, ta memorija je zamišljena da se sastoji od izvesnog broja blokova fiksne dužine, svaki po k reči. To znači da postoji $M = 2^n/k$ blokova u glavnoj memoriji.

Keš memorija se sastoji od C redova. Svaki red sadrži k reči, plus tag od nekoliko bitova. Broj reči u redu se zove veličina reda. Broj redova u kešu je mnogo manji od broja blokova u glavnoj memoriji ($C << M$).



Slika 2. Organizacija keša i glavne memorije

U bilo kom trenutku, neki podskup blokova memorije se nalazi u redovima u kešu. Ako se čita reč u bloku memorije, taj blok se prenosi u jedan od redova keša. S obzirom na to da ima više blokova od redova, pojedinačni red se ne može jedinstveno i trajno dodeliti određenom bloku. Prema tome, svaki red poseduje tag, koji predstavlja deo adrese glavne memorije i koji identifikuje blok koji je trenutno uskladišten u kešu. Organizacija keša je prikazana na slici 3.



Slika 3. Organizacija keš memorije

S obzirom na to da ima manje redova keša od blokova u glavnoj memoriji, potreban je algoritam za preslikavanje blokova glavne memorije u redove keša. Pored toga, potreban je način na koji se određuje koji blok glavne memorije trenutno zauzima red u kešu. Najčešće se koriste tri tehnike preslikavanja:

— Direktno preslikavanje,

— Asocijativno preslikavanje,

— Asocijativno preslikavanje skupa (set-asocijativno preslikavanje).

8.1 Direktno preslikavanje

Ovo je najjednostavnija tehnika preslikavanja gde se svaki blok glavne memorije može preslikati samo u jedan mogući red keša (slika 4). Preslikavanje se izvršava po sledećem obrascu:

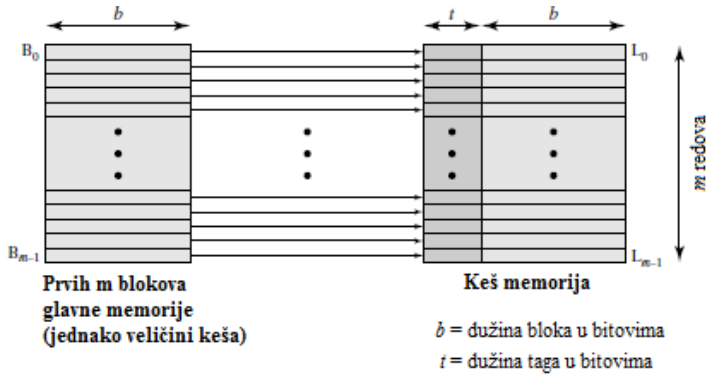
$$i = j \text{ moduo } m,$$

gde je:

i – broj reda u kešu,

j – broj bloka u glavnoj memoriji,

m – broj redova u kešu.

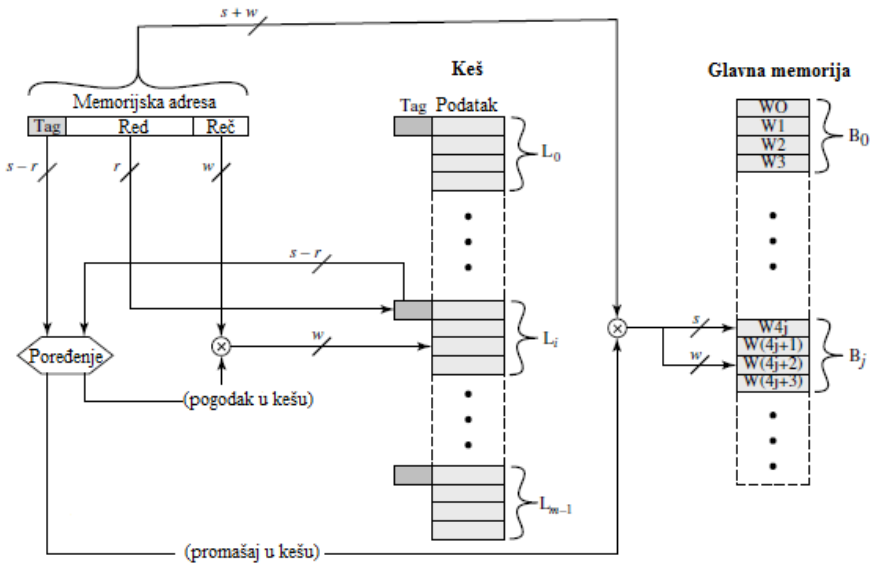


Slika 4. Direktno preslikavanje keša

Funkcija preslikavanja se lako implementira koristeći adrese. Kada se pristupa kešu, svaka adresa glavne memorije može da se posmatra kao da se sastoji iz tri dela.

Najmanje značajnih w bitova identifikuju jedinstvenu reč unutar bloka glavne memorije. To znači da jedan blok glavne memorije sadrži 2^w reči. Ostalih s bitova određuju 2^s blokova glavne memorije. Logika keša interpretira tih s bitova kao tag od s-r bitova i polje reda od r bitova. Polje reda identifikuje jedan od $m = 2^r$ redova keša.

Kod direktnog preslikavanja, prvo se čita red u adresi i tag u kešu koji odgovara tom redu (slika 5). Tag se poredi sa tagom u memorijskoj adresi. Ako su isti, blok se nalazi u kešu i reč se čita. Ako nisu isti, kažemo da je došlo do promašaja u kešu i pristupa se glavnoj memoriji. Blok iz glavne memorije u kojem se nalazi tražena reč se prebacuje u keš memoriju. Tada se čita blok s i reč na adresi w iz tog bloka.



Slika 5. Organizacija keša kod direktnog preslikavanja

Rezime:

- Dužina adrese: $s + w$ bitova,
- Kapacitet memorije: 2^{s+w} reči,
- Veličina bloka (ili reda): 2^w reči,
- Broj blokova u glavnoj memoriji: 2^s,
- Broj redova u keš memoriji: 2^r,
- Veličina taga: $s - r$ bitova.

Tehnika direktnog preslikavanja je jednostavna i jeftina za implementaciju. Njen glavni nedostatak je to što postoji fiksna lokacija u kešu za svaki blok glavne memorije. Ako se dogodi da program stalno traži reči iz dva različita bloka koji se preslikavaju u isti red, onda će se blokovi stalno izbacivati iz keša, a verovatnoća pogotka će biti mala.

U tabeli 1 su dati redovi keša u koje se preslikavaju blokovi operativne memorije

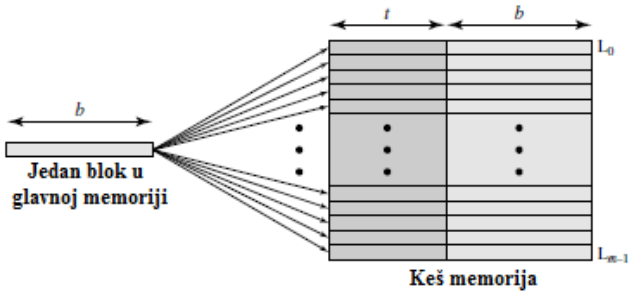
Tabela 1. Redovi keša u koje se preslikavaju blokovi operativne memorije

Red u kešu	**Dodeljeni blokovi glavne memorije**
0	$0, m, 2m, \ldots, 2^s - m$
1	$1, m + 1, 2m + 1, \ldots, 2^s - m + 1$
⋮	⋮
$m - 1$	$m - 1, 2m - 1, 3m - 1, \ldots, 2^s - 1$

8.2 Asocijativno preslikavanje

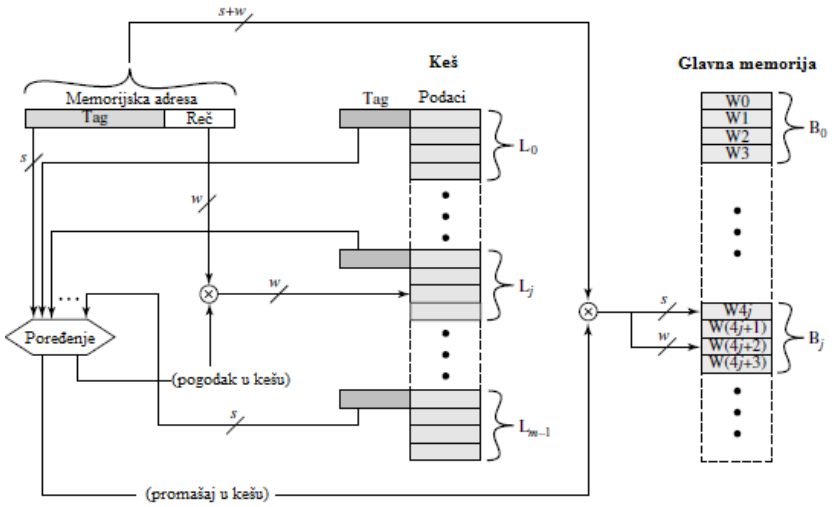
Asocijativno preslikavanje prevazilazi nedostatak direktnog preslikavanja dozvoljavajući svakom memorijskom bloku da se učita u bilo koji red u kešu (slika 6). U tom

slučaju, upravljačka logika keša interpretira memorijsku adresu kao polje za tag i polje reči. Polje za tag jedinstveno identifikuje blok glavne memorije. Da bi odredila da li je blok u kešu, upravljačka logika keša mora istovremeno da ispita tag svakog reda radi pronalaženja podudarnosti.



Slika 6. Asocijativno preslikavanje

Kod asocijativnog preslikavanja postoji fleksibilnost u pogledu toga koji blok treba zameniti kada se novi blok učitava u keš. Osnovni nedostatak ovog preslikavanja su složena elektronska kola koja su potrebna da bi se paralelno ispitivali tagovi svih redova u kešu. Organizacija keša kod asocijativnog preslikavanja je prikazana na slici 7.



Slika 7. Organizacija keša kod asocijativnog preslikavanja

Dakle, kod asocijativnog preslikavanja važi:

- Dužina adrese: $n = s + w$ bita,
- Dužina taga: s bitova,
- Dužina reči: w bitova,
- Kapacitet glavne memorije: 2^n reči,
- Veličina bloka (ili reda): 2^w reči,

– Broj blokova u glavnoj memoriji: 2^s.

8.3 Asocijativno preslikavanje skupa (set-asocijativno preslikavanje)

Ovo je kompromis između direktnog i asocijativnog pristupa, pri čemu do izražaja dolaze njihove prednosti, a umanjuju se njihovi nedostaci. U slučaju asocijativnog preslikavanja skupa, keš je podeljen na v skupova, od kojih se svaki sastoji od k redova. Ovo preslikavanje se vrši prema sledećem obrascu:

$$m = v \cdot k$$
$$i = j \text{ moduo } v$$

gde je:

i – redni broj skupa u keš memoriji,

j – redni broj bloka u glavnoj memoriji,

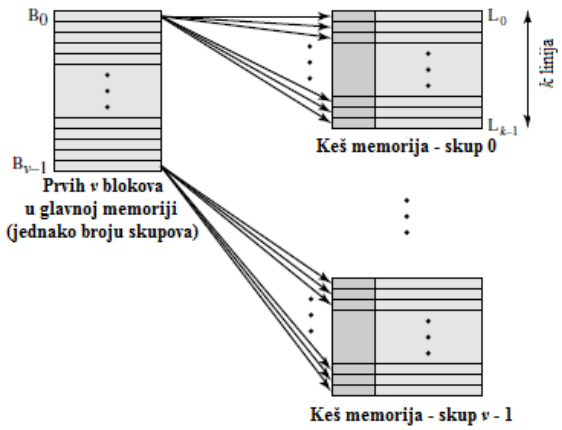
m – broj redova u keš memoriji,

v – broj skupova u keš memoriji,

k – broj redova u svakom skupu.

Ako jedan skup sadrži k redova, onda za preslikavanje kažemo da je k-tostruko asocijativno preslikavanje skupa.

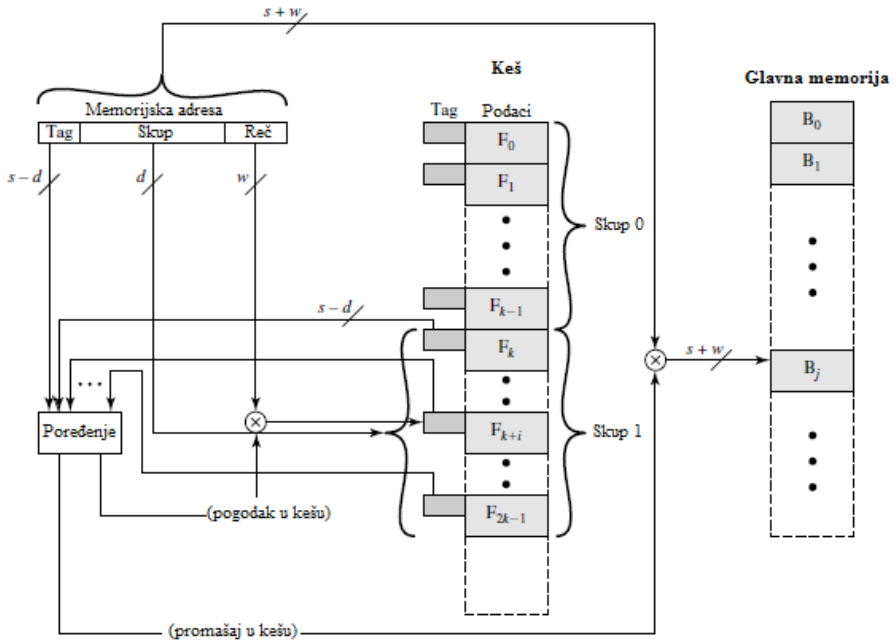
Kod ovog preslikavanja, blok B_j može da se preslika u bilo koji od redova skupa j (slika 8). U tom slučaju, upravljačka logika keša interpretira memorijsku adresu kao tri polja: tag, skup i reč. Pomoću d bitova skupa se određuje jedan od $v = 2^d$ skupova. Jedan od 2^s blokova glavne memorije se određuje pomoću s bitova polja za tag i polja za skup.



Slika 8. Asocijativno preslikavanje skupa

Kod potpuno asocijativnog preslikavanja, tag u memorijskoj adresi je prilično dugačak i mora da se poredi sa tagom od svakog reda u kešu. Kod k-tostrukog asocijativnog

preslikavanja skupa, tag u memorijskoj adresi je mnogo manji i poredi se samo sa *k* tagova unutar jednog skupa (slika 9).



Slika 9. Organizacija keša kod asocijativnog preslikavanja skupa

Kod asocijativnog preslikavanja skupa važi sledeće:

- Dužina memorijske adrese: $n = s + w$ bitova,
- Kapacitet memorije: 2^n reči,
- Veličina bloka: 2^w reči,
- Broj blokova u glavnoj memoriji: 2^s,
- Broj redova u skupu: k,
- Broj skupova: $v = 2^d$,
- Broj redova u keš memoriji: $C = k \cdot v$.

U ekstremnom slučaju kada je $v = m$, $k = 1$, asocijativno preslikavanje skupa postaje direktno preslikavanje, a kada je $v = 1$, $k = m$, postaje asocijativno preslikavanje. Najčešća organizacija ovog preslikavanja je korišćenje dva reda po skupu ($v = m/2$, $k = 2$).

8.4 Promašaj u keš memoriji

Promašaj predstavlja situaciju kada s traženi blok ne nalazi u keš memoriji pa je potrebno njegovo učitavanje iz glavne memorije. Ukoliko imamo mali broj promašaja, procesor će manje vremena trošiti čekajući sporu operativnu memoriju, tj. željene reference će dobijati iz brze keš memorije.

Promašaje u keš memoriji možemo klasifikovati u tipa promašaja:

- Prinudni (*compulsory*) promašaji su promašaji koji se javljaju prilikom prvog prozivanja reference. Oni ne zavise od kapaciteta keš memorije kao ni od tipa koršćenog preslikavanja u keš memoriji, ali zavise od veličine blokova u keš memoriji. Promašaji ovog tipa mogu se izbeći
- Promašaji usled uticaja kapaciteta su gubici na koje ne utiču tip korišćenog preslikavanja kao ni veličina bloka već samo veličina keš memorije.
- Konflikni promašaji nastaju prilikom zamene blikova u keš memoriji i mogu se podeliti u dve podgrupe:
    - Promašaji usled mapiranja su promašaji koji su neizbežni za određenu tehniku preslikavanja
    - Promašaji usled zamene bloka nastaju prilikom izbora bloka koji će biti zamenjen u keš memoriji.

8.5 Keš memorije u više nivoa

Razvojem tehnologije i povećanjem broja tranzistora na čipu, postalo je moguće imati keš na istom čipu kao i procesor: U poređenju sa kešom sa kojim procesor komunicira preko spoljašnje magistrale, keš na čipu smanjuje spoljašnje aktivnosti procesora i na taj način ubrzava vremena izvršenja i povećava ukupnu performansu sistema. Kada se zahtevana instrukcija ili podatak pronađu u kešu na čipu, nije potreban pristup preko magistrate. Zbog kratkih putanja podataka unutar matrice procesora, u poređenju sa dužinama magistrale, pristupi kešu na čipu završavaju se značajno brže čak i od ciklusa magistrale sa nultim stanjima čekanja. Uz to, za vreme tog perioda, magistrala je slobodna za druge prenose.

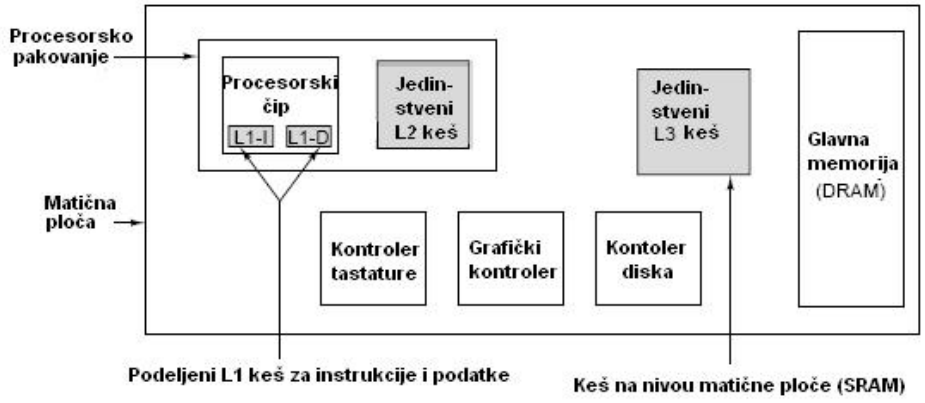
Većina savremenih sistema uključuje i keš na čipu i spoljašnji keš. Najjednostavnija od takvih organizacija je poznata kao keš u dva nivoa, sa internim kešom označenim kao nivo 1 (Ll) i spoljašnjim kešom označenim kao nivo 2 (L2). Razlog za uključivanje keša L2 je sledeći. Ako nema keša L2 i procesor pravi zahtev za pristupom memorijskoj lokaciji koja nije u kešu Ll, onda procesor mora da pristupi DRAM ili ROM memoriji preko magistrale. Zahvaljujući tipično maloj brzini magistrale i dugom vremenu pristupa, to rezultuje slabom performansom. S druge strane, ako se koristi keš L2 SRAM (statička RAM memorija), onda često promašena informacija može brzo da se pozove. Ako je SRAM memorija dovoljno brza da odgovara brzini procesora, onda podatku može da se pristupi putem transakcije sa nultim vremenom čekanja, što je najbrža vrsta prenosa preko magistrale.

Napomenimo dva svojstva savremenog dizajna keš memorija u više nivoa. Prvo, za keš L2 izvan čipa, mnoge konstrukcije ne koriste sistemsku magistralu kao putanju za prenos

između keša L2 i procesora, nego posebnu putanju za podatke da bi se smanjilo opterećenje sistemske magistrale.

Drugo, sa smanjivanjem procesorskih komponenata, veliki je broj procesora sada ima keš L2 u pakovanju procesorskog čipa, što poboljšava performansu.

Moguće uštede zahvaljujući upotrebi keša L2 zavise od verovatnoća pogodaka, kako u kešu Ll, tako i u kešu L2. Više studija je pokazalo da upotreba keš memorije drugog nivoa zaista poboljšava performansu, ali, upotreba keš memorija u više nivoa komplikuje sva pitanja projektovanja, uključujući veličinu, algoritam zamene i politiku upisivanja.



Slika 10. Keš memorija u više nivoa

Sa povećanjem raspoloživosti površine na čipu za keš, u većini savremenih mikroprocesora keš L2 se premešta na procesorski čip, a dodaje se keš L3. Keš L3 je prvobitno bio pristupačan preko spoljašnje magistrale, a u novije vreme, u većini mikroprocesora i L3 keš je ugrađen u čipu.