



## CS120 - ORGANIZACIJA RAČUNARA

Keš memorija

Lekcija 06

PRIRUČNIK ZA STUDENTE

# CS120 - ORGANIZACIJA RAČUNARA

## Lekcija 06

### ***KEŠ MEMORIJA***

- ✓ Keš memorija
- ✓ Poglavlje 1: Ubrzavanje performansi računara
- ✓ Poglavlje 2: Direktno mapirana keš memorija
- ✓ Poglavlje 3: Asocijativno mapirana keš memorija
- ✓ Poglavlje 4: Set-asocijativno mapirana keš memorija
- ✓ Poglavlje 5: Višenivoovska keš memorija
- ✓ Poglavlje 6: Pokazne vežbe
- ✓ Poglavlje 7: Zadaci za samostalni rad
- ✓ Poglavlje 8: Domaći zadatak
- ✓ Zaključak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

## ✓ Uvod

# UVOD

### *Uvod u lekciju #6*

U ovoj lekciji biće najpre reči o razlozima uvođenja keš memorije u hijerarhijski sistem memorija u računaru.

Opisaće se funkcije mapiranja, uz primere za direktno-mapiranu, asocijativno-mapiranu i set-asocijativno-mapiranu keš memoriju.

Svaka funkcija mapiranja ima dovoljno primera da opiše postupak organizacije blokova keš operativne memorije i linija keš memorije.

Savremeni procesori koriste keš memoriju sa više nivoa, tako da sledeći objekat učenja daje osvrt na višenivoovsku keš memoriju.

## ▼ Poglavlje 1

# Ubrzavanje performansi računara

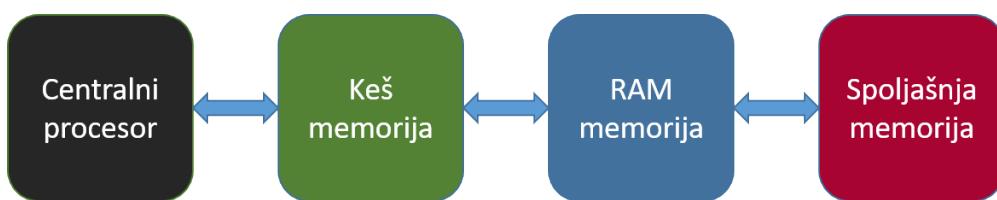
## HIJARARHISKI KONCEPT MEMORIJA U RAČUNARU

*Procesor pristupa informacijama, naredbama i podacima, koje su samo u keš memoriji.*

Hijerarhijski koncept memorije podrazumeva nivo **brze memorije** ili **keš memorije** (en. **cache**) kojoj procesor pristupa u jednom taktu, i nivoe sporije memorije koja je znatno većeg kapaciteta.

Procesor pristupa informacijama, naredbama i podacima, koje su samo u keš memoriji.

Ukoliko informacija nije u keš memoriji, potrebno ju je prebaciti iz sporije u brzu memoriju.



Slika 1.1 Organizacija memorijskog sistema računara. [Izvor: Autor]

## KEŠ MEMORIJA

*Keš memorija radi na principu prostornog i vremenskog lokaliteta*

Što procesor češće nalazi potrebne informacije u brzoj memoriji, ređe je potrebno iste prebacivati iz spore u brzu memoriju te sistem radi bliže maksimalno mogućoj brzini obrade.

Princip lokaliteta (en. **principle of locality**), jeste činjenica da program u nekom vremenskom intervalu pristupa relativno uskom memorijском području.

Lokalitet može biti **prostorni** (en. **spatial**) i **vremenski** (en. **temporal**).

**Prostorni lokalitet** podrazumeva da ukoliko se u jednom trenutku pristupilo jednoj memorijskoj lokaciji da je velika verovatnoća da će u sledećim trenucima pristupati njoj susednim lokacijama.

**Vremenski lokalitet** je posledica činjenice da ukoliko se pristupi jednoj memorijskoj lokaciji da je velika verovatnoća ponovnog pristupa istoj unutar kratkog vremenskog intervala.

Uz pojam lokaliteta vezuje se **radni skup memorijskih lokacija** (en. **working set of memory locations**) koji se odnosi na skup memorijskih lokacija kojima se pristupa u određenom vremenskom intervalu.

Kod većine programa radni skup se relativno sporo menja s vremenom. Sistemska adresa, npr. adresa iz procesora, dovodi se na *jedinicu za upravljanje memorijom* (en. **Memory Management Unit, MMU**).

**Rad brze memorije transparentan je programeru.**

Program generiše efektivnu adresu (nazvana sistemska adresa) i definiše operaciju (čitanje ili pisanje).

Memorijski sistem mora realizovati ovu operaciju nezavisno da li je informacija u primarnoj ili samo u sekundarnoj memoriji. Način realizacije memorijske transakcije nije vidljiv programu i programeru.

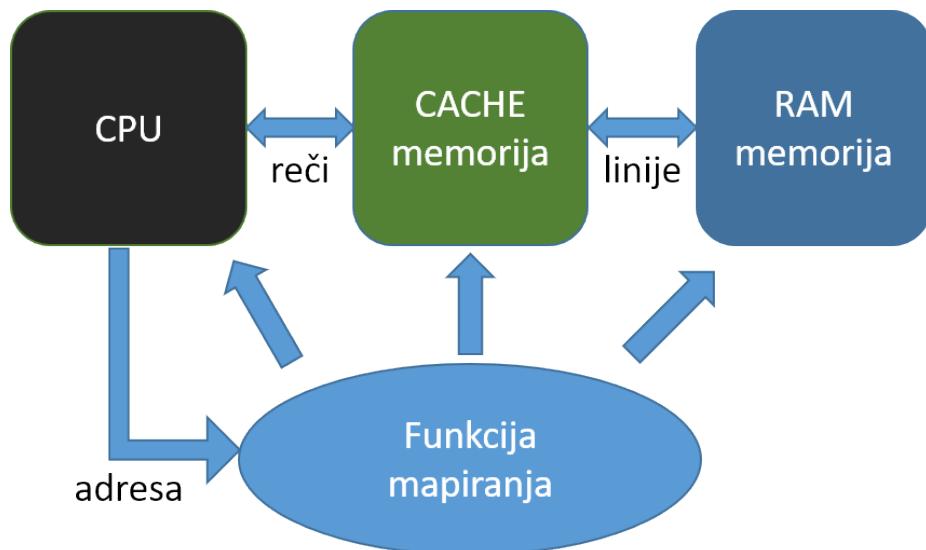
## FUNKCIJE PRESLIKAVANJA

*Između procesora i primarne memorije prenose se reči dok se između primarne i sekundarne memorije prenose linije (blokovi) reči.*

Između procesora i primarne memorije prenose se reči dok se između primarne i sekundarne memorije prenose linije (blokovi) reči.

Funkcije preslikavanja odgovorne su za funkcioniranje višenivoovske memorije. Zbog brzine rada ove funkcije su sklopovski realizovane i određuju sledeće:

- *Strategiju unosa linije* - gde u brzu memoriju treba pohraniti liniju iz glavne memorije,
- *Strategiju zamene* - koju liniju iz brze memorije zameniti ako adresirana linija nije u brzoj memoriji (en. **cache miss**),
- *Strategiju čitanja i pisanja* - kako izvoditi operacije čitanja i pisanja ukoliko je linija u brzoj memoriji (en. **cache hit**) ili nije u njoj.



Slika 1.2 Funkcije mapiranja keš memorije [Izvor: Autor]

## ADRESIRANJE OPERATIVNE MEMORIJE

*CPU može generisati određeni broj jedinstvenih adresa koje ukazuju na svaki bajt.*

Kroz primer biće objašnjeno adresiranje memorije.

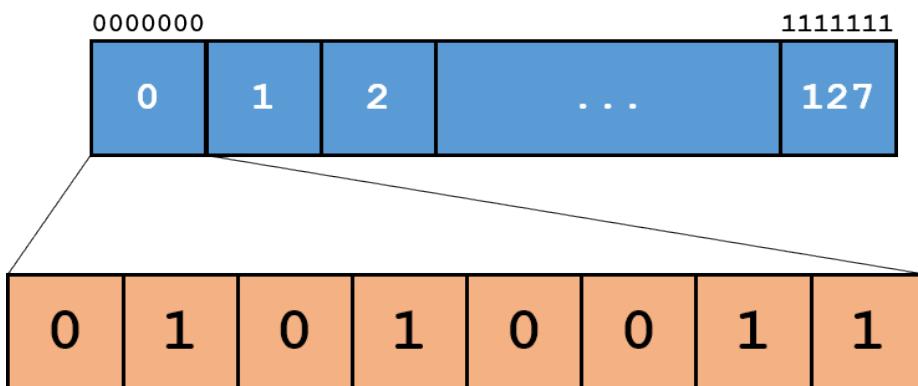
### Primer #1:

Neka je operativna memorija veličine 128 B i da hardver može pristupiti pojedinačnim bajtovima.

CPU može generisati 128 jedinstvenih adresa koje ukazuju na svaki bajt.

Ovo su fizičke adrese i one su u binarnom formatu. Pošto je memorija 128 bajtova, a to je  $2^7$ , treba nam 7 bitova da adresiramo svaku adresu.

Kada procesor prebacuje podatke iz operativne memorije u keš memoriju, ne prebacuje samo jedan bajt, **već više bajtova**, koje se zovu **blokovi**.



Slika 1.3 Pristupanje pojedinačnim bajtovima [Izvor; Autor]

## PREDSTAVLJANJE OPERATIVNE MEMORIJE KAO SKUP BLOKOVA

*Uместо predstavljanja kroz niz blokova, memoriju predstavićemo kroz matricu blokova zbog lakšeg adresiranja.*

Predstavićemo operativnu memoriju kao skup blokova. Jedan memorijski blok ima 8 bajtova. Ukupno imamo 16 blokova.

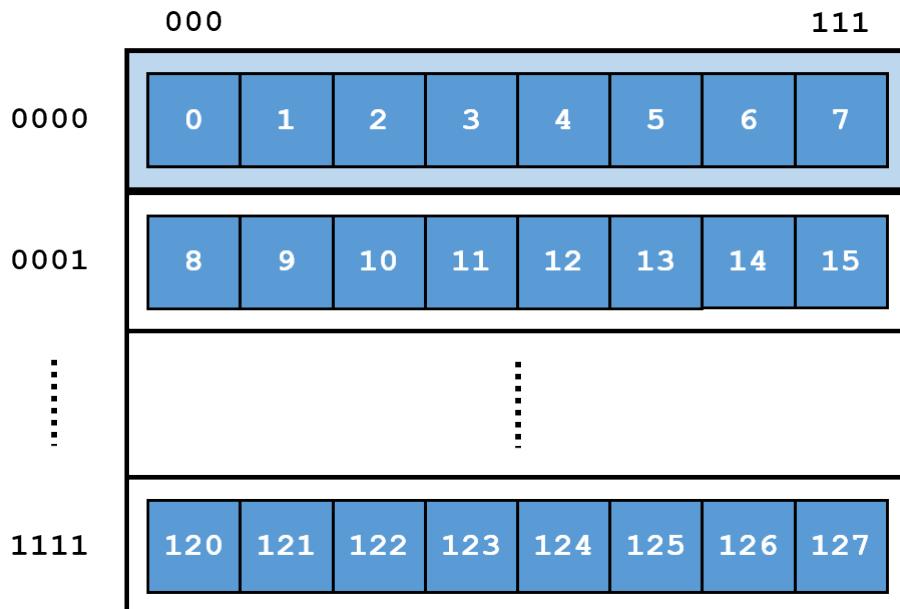
Veličinu memorije (broj bajtova) podelimo sa brojem bajtova po bloku.

$$128/8 = 16.$$

Svaki blok označavamo indeksom. Ukupno ih imamo 16, pa će prvi indeks biti 0, a poslednji 15. Binarno,  $16 = 2^4$ , pa su nam potrebna 4 bita za indeksiranje blokova.

Svaku kolonu takođe možemo označiti, i to *pomerajem* (en. **offset**). Ukupno imamo 8 kolona ( $2^3$ ), pa su nam potreba 3 bita za pomeraj.

CPU može adresirati bajt 7-bitnom fizičkom adresom.



Slika 1.4 Predstavljanje operativne memorije kao skup blokova. [Izvor: Autor]

## ADRESA KONKRETNOG BAJTA

*Od ukupnog broja bitova za adresiranje memorije, deo be biti indeks bloka, a deo offset bloka.*

CPU može adresirati bajt 7-bitnom fizičkom adresom, ali sada je predstavljanje kroz matricu umesto kroz niz.

Od tih sedam bitova, su prva 4 bita za *indeks bloka* (**BI**), a preostala 3 bita za *offset bloka* (**BO**).

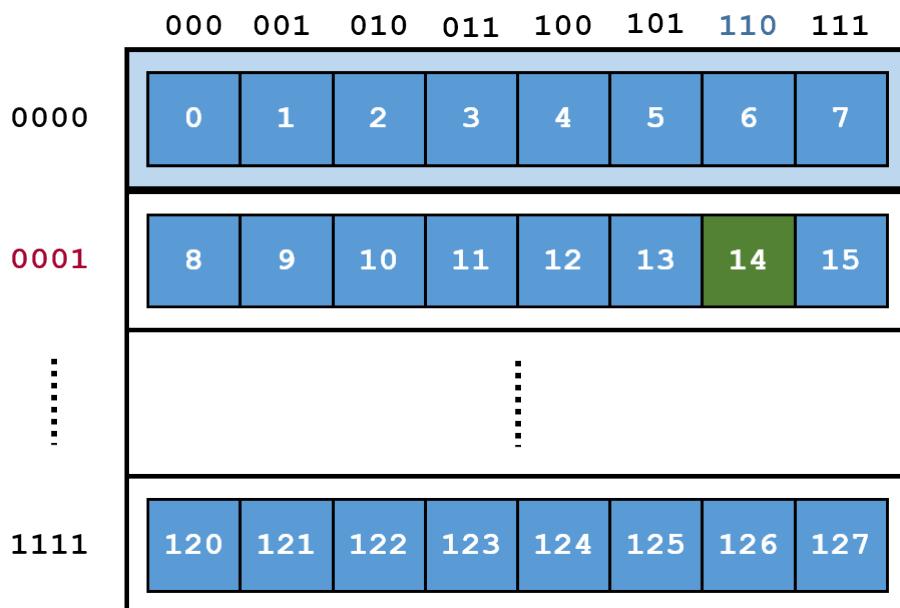
**Primer:**

CPU traži sledeću fizički adresu (en. physical address, PA):

**PA: 0001110**

**CPU:**  
**PA: 0001110**

Slika 1.5 Zahtev za fizičkom adresom [Izvor: Autor]



Slika 1.6 Pronalaženje fizičke adrese kroz matricu blokova [Izvor: Autor]

## ADRESIRANJE KEŠ MEMORIJE

*U RAM memoriji imamo blokove, u cache memoriji imamo linije!*

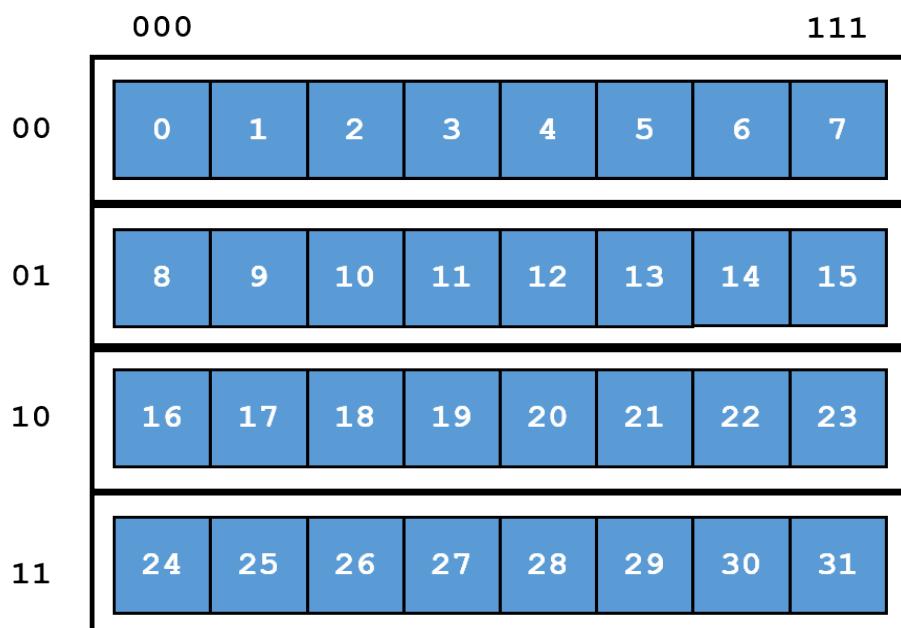
Prateći prethodni primer, predstavićemo cache memoriju od 32 B kao skup blokova.

Jedan memorijski blok isto ima 8 bajtova. Ukupno imamo 4 linije.

**Napomena:**

***U RAM memoriji imamo blokove, u cache memoriji imamo linije!***

***Veličina bloka i linije su jednake!***



Slika 1.7 Adresiranje keš memorije kroz linije [Izvor: Autor]

## ✓ Poglavlje 2

### Direktno mapirana keš memorija

#### ORGANIZACIJA DIREKTNO-MAPIRANE KEŠ MEMORIJE

*Kod direktnog mapiranja određeni blok uvek ide u određenu liniju u keš memoriji!*

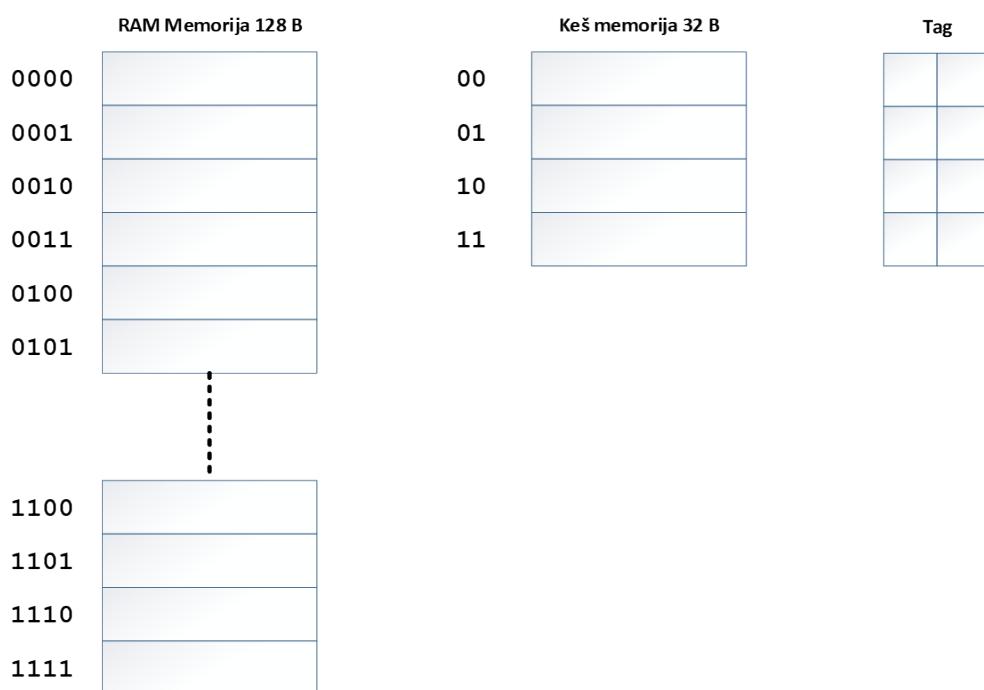
Prvi tip mapiranja keš memorije jeste direktno-mapirana keš memorija.

Kod direktnog mapiranja određeni blok uvek ide u određenu liniju u keš memoriji! Ovo je određeno **LSB** (en. **least significant bit**) indeksom bloka.

Kod direktno-mapirane keš memorije pored svake linije keš memorije imamo i određene tag bitove, koji nam označavaju da znamo koji je tačno blok operativne memorije mapiran u tu liniju keš memorije.

Na slici je predstavljena operativna memorija od 128B i keš memorija sa 32B koja poseduje 2 tag bita po liniji.

Na primeru biće objašnjeno kako se vrši mapiranje i zašto imamo tačno dva tag bita po liniji.



Slika 2.1 Organizacija direktno-mapirane keš memorije [Izvor: Autor]

## ODREĐIVANJE TAG BITOVA

*Tag bitovi određuju koji su konkretno blokovi prebačeni u liniju.*

**Primer:**

Odrediti broj tag bitova keš memorije od 32B ako je operativna memorija 128B

**Rešenje:**

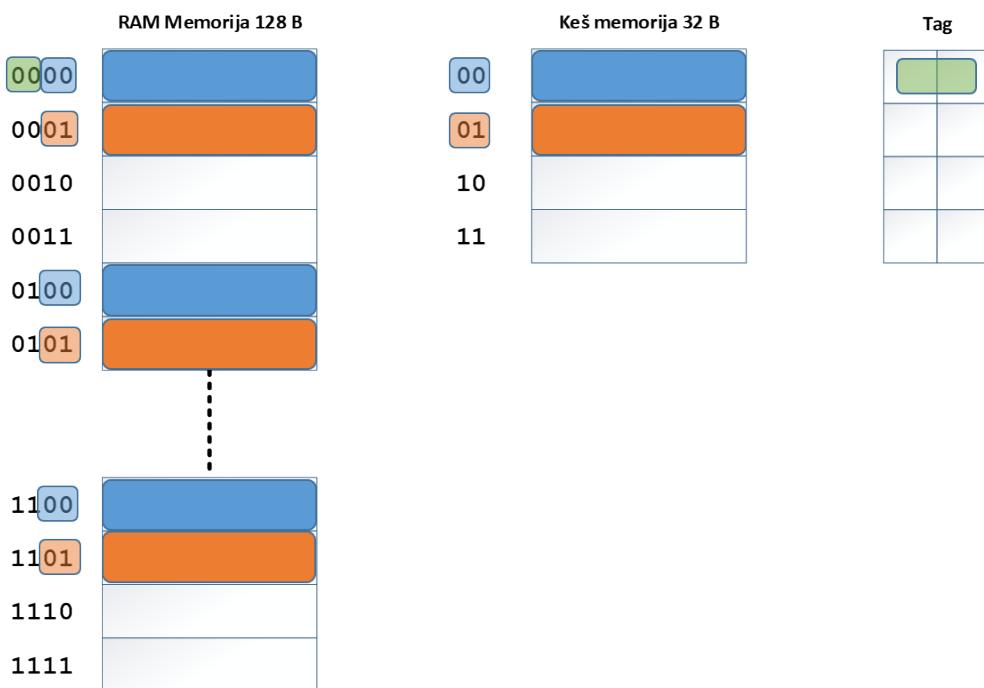
Broj tag bitova određuje se kad od indeksa bloka (BI) oduzmememo indeksa keš linije (CI):

$$BI - CI = 4 - 2 = 2 \text{ bita.}$$

U ove bitove stavljamo prva MSB (en. **most significant bit**) iz indeksa bloka.

Budući da kod direktno-mapirane keš memorije određeni blok operativne memorije uvek ide u određenu liniju u keš memoriji, a keš memorija je znatno manjeg kapaciteta, onda treba znati koji blok je u kojoj liniji, i tome služe tagovi.

Na slici se vidi da svaki plavi blok može ići u plavu liniju, narandžasti blok u narandžastu liniju itd., a imamo samo jednu plavu i jednu narandžastu liniju. Tag bitovi određuju koji su to bitovi u pitanju.

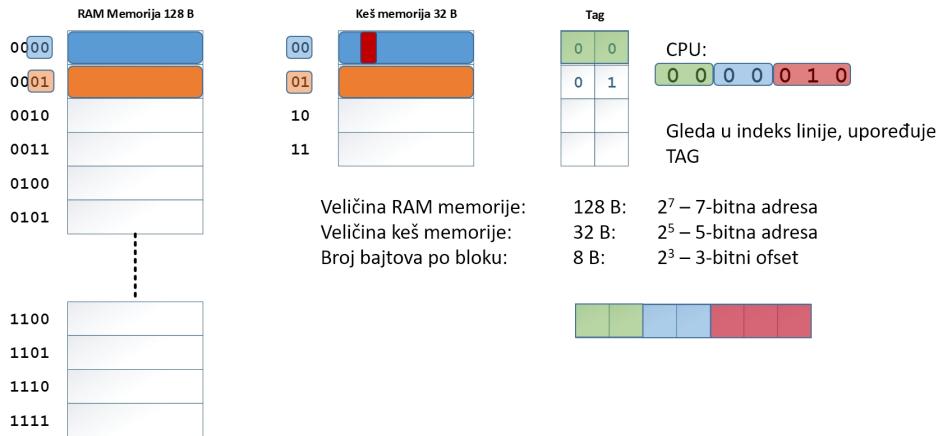


Slika 2.2 direktno-mapirana keš memorija [Izvor: Autor]

## RAČUNANJE FIZIČKE ADRESE

*Fizička adresa računa se iz delova: tag bitovi, indeks linije, pomeraj*

Procesor i dalje zahteva 7-bitnu adresu (to je određeno kapacitetom operativne memorije). Međutim, u direktno-mapiranoj keš memoriji ova adresa se računa iz delova (od MSB ka LSB) tag bitova - 2 bita (zeleni bitovi), indeksa linije (reda) - 2 bita (plavi) i pomeraja ili ofseta (reči) - 3 bita (crveni).



Slika 2.3 Računanje fizičke adrese kroz direktno-mapiranu keš memoriju [Izvor: Autor]

## OSOBINE DIREKTNO-MAPIRANE KEŠ MEMORIJE

*Najveća prednost direktno-mapirane keš memorije jeste mala potrošnja energije*

### Stavljanje blok u keš:

Kod direktnog mapiranja određeni blok uvek ide u određenu liniju u keš memoriji! Ovo je određeno indeksom linije.

### Prednosti:

- Direktno mapiranje je najjednostavnije.
- Zahteva najmanje el. energije jer ne pretražuje sve keš linije.
- Zahteva jednostavni hardver jer samo jedna keš linija se u jednom trenutku poredi.

### Nedostaci:

- Mala brzina cache hit-a, jer postoji samo jedna keš linija po mapiranju.
- Svaki put kad se ubacuje nova vrednost, stara se izbacuje.

## DIREKTNO-MAPIRANA KEŠ MEMORIJA: PRIMER #1

*Postavka prvog zadatka za direktno-mapiranu keš memoriju.*

### Primer #1 (7 minuta)

Računar poseduje operativnu memoriju veličine 4 kB i keš memoriju veličine 64 B.

Veličina bloka u memoriji je 8 B.

Prikazati organizaciju glavne memorije, keša i polje fizičke adrese, ako je u pitanju direktno preslikavanje keša.

### Rešenje:

Kapacitet RAM memorije  $C_{RAM} = 4 \text{ kB} = 4096 \text{ B} = 2^{12} \text{ B}$

Kapacitet keš memorije  $C_{CACHE} = 64 \text{ B} = 2^6 \text{ B}$

Dužina memorijske adrese A =  $\log_2(C_{RAM}) = 12$

Veličina bloka B = 8 B =  $2^3 \text{ B}$

Broj blokova RAM memorije  $N_{MB} = C_{RAM} / B = 4096 / 8 = 512 = 2^9$

Broj linija keš memorije  $N_{CB} = C_{CACHE} / B = 64 / 8 = 8 = 2^3$

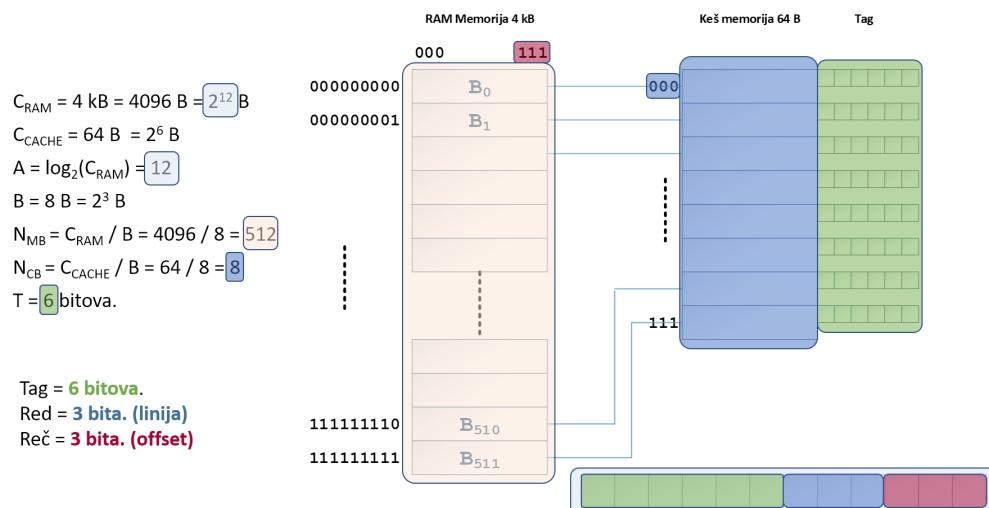
Kod direktnog preslikavanja, broj tag bitova T dobijamo kada od indeksa bloka ( $\log_2(N_{MB})$ ) oduzmememo indeksa keš linije ( $\log_2(N_{CB})$ )

T = 9 - 3 = 6 bitova.

Alternativno, T =  $\log_2(N_{MB}/N_{CB})$

## PRIMER #1: PRIKAZ ORGANIZACIJE MEMORIJE

### *Prikaz organizacije memorije za primer #1*



Slika 2.4 Organizacija memorije kod primera #1 [Izvor: Autor]

## DIREKTNO-MAPIRANA KEŠ MEMORIJA: PRIMER #2

*Postavka drugog zadatka za direktno-mapiranu keš memoriju.*

### Primer #2 (5 minuta)

Računar poseduje memorijski podsistem koji se sastoji iz operativne memorije i keša sa direktnim preslikavanjem. Odrediti kapacitet glavne memorije i keša, ako su polja fizičke adrese prikazana na slici.



Slika 2.5 Polja fizičke adrese za primer #2 [Izvor: Autor]

**Rešenje:**

$$\text{Kapacitet RAM memorije } C_{RAM} = 2^n B = 2^{32} B = 2^2 * 2^{30} B = 4 \text{ GB}$$

$$\text{Veličina bloka } B = 2^w * 2^{12} B = 4 \text{ kB}$$

$$\text{Broj blokova RAM memorije } N_{MB} = C_{RAM} / B = 2^{32} / 2^{12} = 2^{20}$$

$$\text{Broj linija u keš memoriji } N_{CB} = 2^r = 2^9 = 512$$

$$\text{Kapacitet keš memorije } C_{CACHE} = N_{CB} * B = 2^9 * 2^{12} = 2^{21} = 2 \text{ MB}$$

## ▼ Poglavlje 3

# Asocijativno mapirana keš memorija

## SVOJSTVA ASOCIJATIVNO-MAPIRANE KEŠ MEMORIJE

*Asocijativno-mapirana keš memorija je najfleksibilnija funkcija mapiranja.*

Asocijativno-mapirana keš memorija ima svojstvo da se svaki memorijski blok može smestiti u proizvoljnu liniju u keš memoriji!

### Prednosti:

- Asocijativno mapiranje je najfleksibilnije.
- Postavljanje bloka u liniju pruža bolji odnos **cache hit/ cache miss**.
- Pruža mogućnost korišćenja više algoritama ukoliko se desi cache miss.

### Nedostaci:

- Sporo postavljanje bloka u liniju zbog većeg broja iteracija.
- Zahteva više el. energije jer se pretražuje ceo keš blok.

## ORGANIZACIJA ASOCIJATIVNO-MAPIRANE KEŠ MEMORIJE

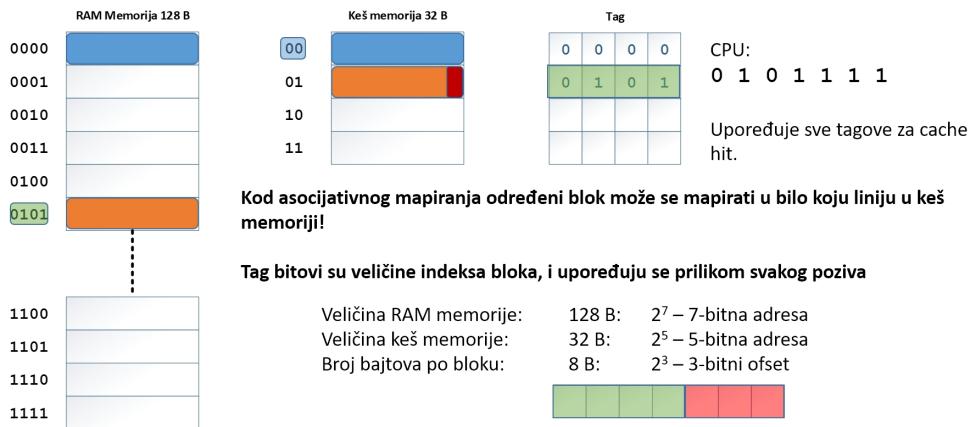
*U polju fizičke adrese ne postoji indeks linije, već samo tag bitovi i bitovi za pomeraj.*

Kod asocijativnog mapiranja određeni blok može se mapirati u bilo koju liniju u keš memoriji!

Tag bitovi su **veličine indeksa bloka**, i upoređuju se prilikom svakog poziva.

Kao i kod osnovnog primera za direktno-mapirane keš memorije, naš sistem imaće 128B operativne memorije i 32B keš memorije.

Možemo primetiti da u fizičkoj adresi **ne postoji indeks linije**, već samo tag (zeleni bitovi) i pomeraj (crveni bitovi)



Slika 3.1 Organizacija asocijativno-mapirane keš memorije [Izvor: Autor]

## ASOCIJATIVNO-MAPIRANA KEŠ MEMORIJA: PRIMER #3

*Postavka zadatka za asocijativno-mapiranu keš memoriju.*

### Primer #3 (7 minuta)

Računar poseduje operativnu memoriju kapaciteta 2 kB i keš memoriju kapaciteta 256 B koji poseduje asocijativno preslikavanje.

Nacrtati organizaciju operativne i keš memorije i odrediti polje fizičke adrese, ako je veličina bloka u operativnoj memoriji 64 B.

#### Rešenje:

$$\text{Kapacitet RAM memorije CRAM} = 2 \text{ kB} = 2048 \text{ B} = 2^{11} \text{ B}$$

$$\text{Kapacitet keš memorije C CACHE} = 256 \text{ B} = 2^8 \text{ B}$$

$$\text{Dužina memoriske adrese A} = \log_2(\text{C RAM}) = 11$$

$$\text{Veličina bloka B} = 64 \text{ B} = 2^6 \text{ B}$$

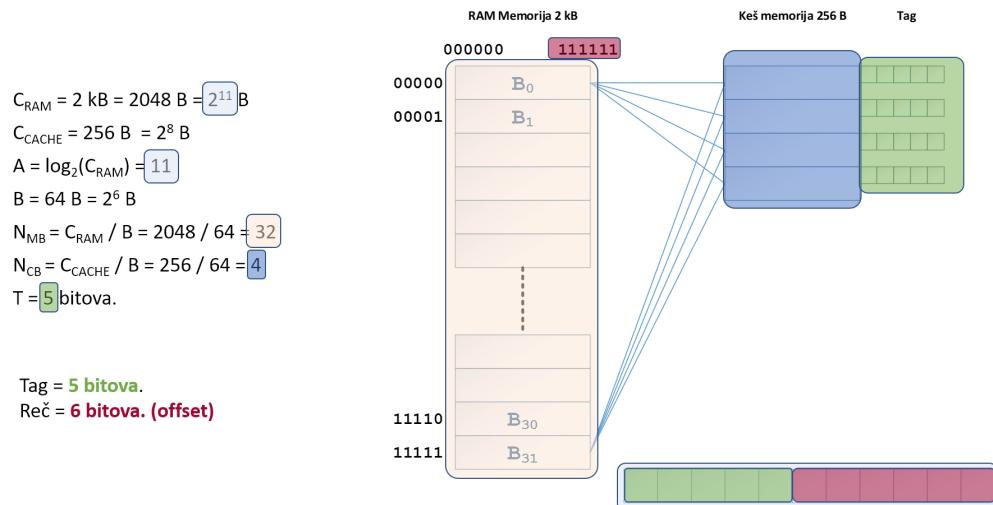
$$\text{Broj blokova RAM memorije N MB} = \text{C RAM} / \text{B} = 2048 / 64 = 32 = 2^5$$

$$\text{Broj linija keš memorije NCB} = \text{C CACHE} / \text{B} = 256 / 64 = 4 = 2^2$$

Kod asocijativnog preslikavanja, broj tag bitova je jednak broju indeksa bloka, tj.  $\log_2(N_{MB}) = T = 5$

## PRIMER #3: PRIKAZ ORGANIZACIJE MEMORIJE

*Prikaz organizacije memorije za primer #3*



Slika 3.2 Organizacija memorije kod primera #3 [Izvor: Autor]

## ▼ Poglavlje 4

# Set-asocijativno mapirana keš memorija

## SVOJSTVA SET-ASOCIJATIVNO-MAPIRANE KEŠ MEMORIJE - KOPIJA

*Set-asocijativno-mapirana keš memorija neće najefikasnije koristiti sve dostupne keš linije.*

Set-asocijativno-mapirana keš memorija ima svojstvo da se svaki memorijski blok može smestiti u **proizvoljnu liniju u unutar određenog skupa** u keš memoriji!

### Prednosti:

- Set-asocijativno mapiranje predstavlja balans između direktnog i asocijativnog mapiranja.
- Pruža mogućnost korišćenja više algoritama ukoliko se desi cache miss.

### Nedostaci:

- Postavljanje bloka u liniju neće najefikasnije koristiti sve dostupne keš linije.

## ORGANIZACIJA SET-ASOCIJATIVNO-MAPIRANE KEŠ MEMORIJE

*U polju fizičke adrese postoji indeks skupa, tag bitovi i bitovi za pomeraj.*

Kod set-asocijativnog mapiranja određeni blok može se mapirati u bilo koju liniju unutar tačno određenog skupa (seta).

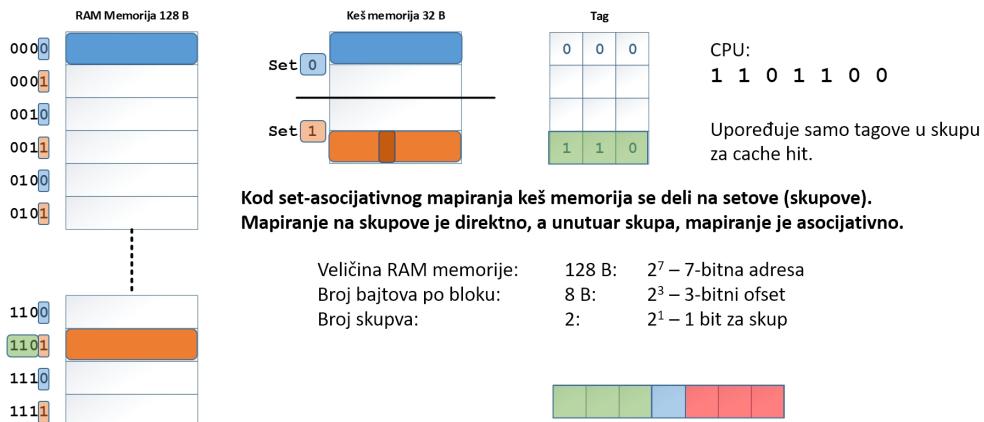
Kod set-asocijativnog mapiranja keš memorija se deli na setove (skupove).

**Mapiranje na skupove je direktno, a unutar skupa, mapiranje je asocijativno.**

Kao i kod osnovnog primera za direktno-mapirane keš memoriju, naš sistem imaće 128B operativne memorije i 32B keš memorije.

Možemo primetiti da u fizičkoj adresi ne postoji indeks linije, ali postoji **indeks skupa** (plavi bit).

Naravno, ostaju tag bitovi (zeleni bitovi) i bitovi za pomeraj (crveni bitovi)



Slika 4.1 Organizacija set-asocijativno-mapirane keš memorije [Izvor: Autor]

## SET-ASOCIJATIVNO-MAPIRANA KEŠ MEMORIJA: PRIMER #4

*Postavka zadatka za set-asocijativno-mapiranu keš memoriju.*

### Primer #4 (7 minuta)

Računar poseduje operativnu memoriju kapaciteta 4 kB podeljenu u blokove veličine 16 B, a keš je veličine 512 B i koristi 8-struko ( $k=8$ ) asocijativno preslikavanje skupa.

Prikazati organizaciju operativne i keš memorije, kao i format fizičke adrese.

**Napomena:**

***k označava broj linija po skupu, a ne broj skupova!***

### Rešenje:

$$\text{Kapacitet RAM memorije } C_{\text{RAM}} = 4 \text{ kB} = 4096 \text{ B} = 2^{12} \text{ B}$$

$$\text{Kapacitet keš memorije } C_{\text{CACHE}} = 512 \text{ B} = 2^9 \text{ B}$$

$$\text{Dužina memorijske adrese } A = \log_2(C_{\text{RAM}}) = 12$$

$$\text{Veličina bloka } B = 16 \text{ B} = 2^4 \text{ B}$$

$$\text{Broj blokova RAM memorije } N_{\text{MB}} = C_{\text{RAM}} / B = 4096 / 16 = 256 = 2^8$$

$$\text{Broj linija keš memorije } N_{\text{CB}} = C_{\text{CACHE}} / B = 512 / 16 = 32 = 2^5$$

$$\text{Broj skupova keš memorije } v = N_{\text{CB}} / k = 32 / 8 = 4 = 2^2$$

Kod set-asocijativnog preslikavanja, broj bitova za skup dobija se kao

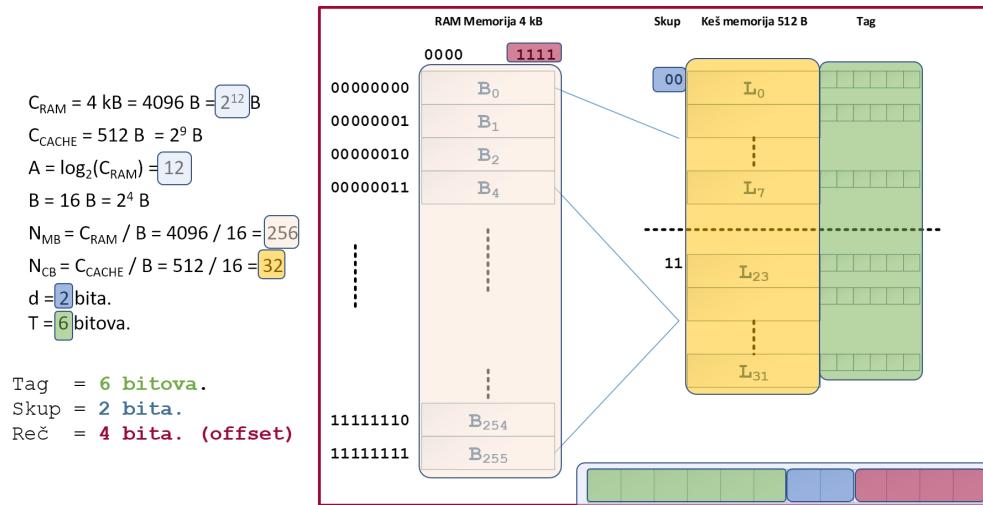
$$d = \log_2(v) = 2$$

Broj tag bitova T dobijamo kada od indeksa bloka ( $\log_2(N \text{ MB})$ ) oduzmememo broj bitova za skup ( $\log_2(v)$ )

$$T = 8 - 2 = 6 \text{ bitova.}$$

## PRIMER #4: PRIKAZ ORGANIZACIJE MEMORIJE

*Prikaz organizacije memorije za primer #4*



Slika 4.2 Organizacija memorije kod primera #4 [Izvor: Autor]

## ▼ Poglavlje 5

# Višenivoovska keš memorija

## NIVOI KEŠ MEMORIJE

*U savremenim procesorima postoji više nivoa keš memorije*

Dosadašnja razmatranja odnosila su se na najbrži tip keš memorije, tzv. Keš prvog nivoa (en. Level 1 cache, L1 cache).

L1 cache (nazvana i interna brza memorija) ima značajan učinak na karakteristike sistema, ali zbog relativno malog kapaciteta učestali pristupi glavnoj memoriji ponovo su ograničavajući faktor u radu sistema. U smislu poboljšanja karakteristika sistema, projektanti ubacuju najpre jedan, pa posle dva nivoa brze memorije koji se smeštaju između procesora (njegove interne brze memorije) i glavne memorije.

L2 i L3 keš memorija, nazvana i eksterna brza memorija ima funkciju provere da li je informacija kojoj se pristupa upisana u njoj kao i način prenosa bloka informacija iz i u glavnu memoriju.

Prema načinu realizacije razlikuje se:

- Paralelni spoj (en. look-aside)
- Serijski spoj (en. look-trough).



Slika 5.1 Nivoi keš memorije [Izvor: Autor]

## PARALELNI SPOJ KEŠ MEMORIJE (LOOK-ASIDE CACHE)

*U slučaju promašaja, tražena linija se prebacuje iz glavne memorije (RAM) što dovodi do usporenja rada sistema.*

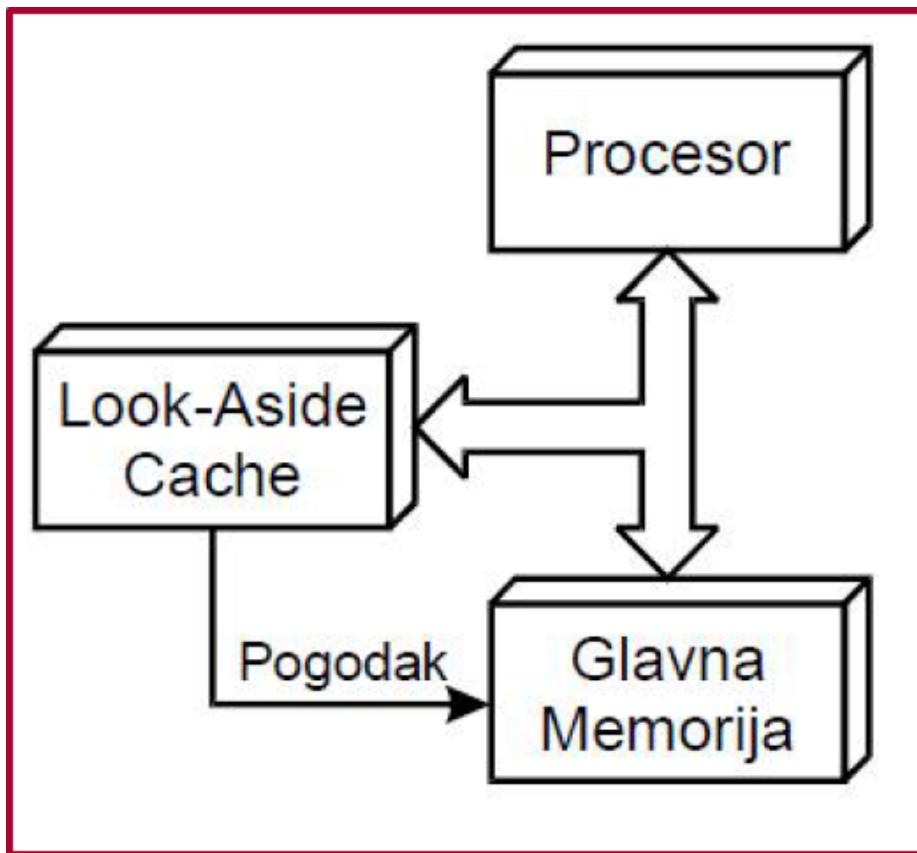
Keš memorija smeštena je paralelno procesorskoj spoljašnjoj magistrali i prati aktivnosti procesora.

Kada procesor pristupa spoljašnjoj memoriji (desio se cache miss) L2 cache proverava da li je tražena informacija pohranjena u njoj ili ne.

Ako jeste, prebacuje se linija iz L2 u L1 cache.

U slučaju promašaja, tražena linija se prebacuje iz glavne memorije (RAM) što dovodi do usporenja rada sistema.

Paralelno sa prebacivanjem informacije u L1 cache, ista se upisuju u L2 cache. Na prvi pogled izgleda da je L2 keš memorija nepotrebna jer se identični sadržaj nalazi i u L1 keš memoriji. Uvek treba voditi računa da je L2 keš memorija znatno većeg kapaciteta od L1 memorije.



Slika 5.2 Look-Aside Cache memorija [Izvor: Autor]

## SERIJSKI SPOJ KEŠ MEMORIJE (LOOK-THROUGH CACHE)

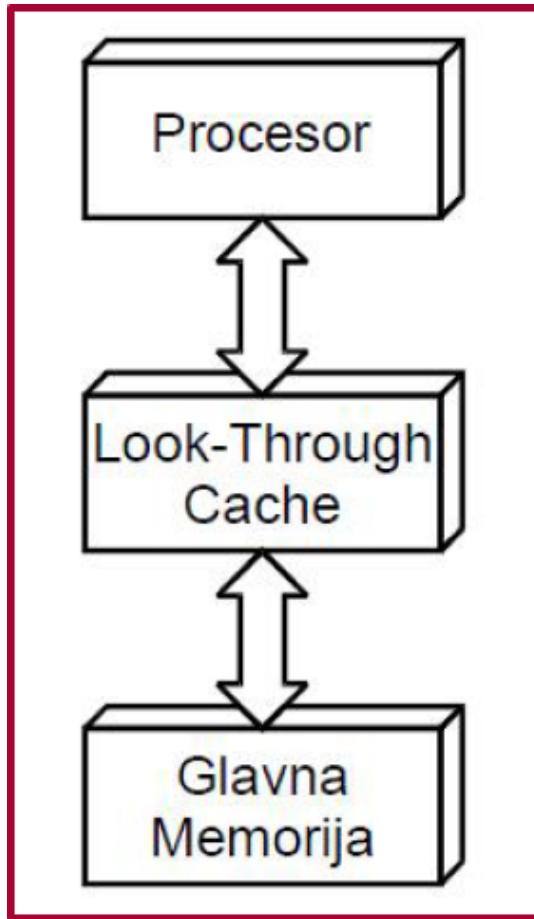
*Zadatak L2 memorije je da snabde L1 keš memoriju memoriju informacijom koja nije u njoj pohranjena.*

Za razliku od paralelnog spoja, u serijskom spoju L2 keš memorija je smeštena između procesora i operativne memorije.

Zadatak L2 memorije je da snabde L1 keš memoriju memoriju informacijom koja nije u njoj pohranjena.

U slučaju **cache miss** (informacija nije pohranjena u L2 memoriji, ali je u operativnoj memoriji) linija se prvo preko sistemske magistrale iz operativne memorije prebacuje u L2 brzu memoriju i istovremeno kroz nju prosleđuje L1 keš memoriji.

Nedostatak ovakvog rešenja je unošenje dodatnog kašnjenja koje je posledica ispitivanja da li je potrebna informacija u L2 brzoj memoriji.



Slika 5.3 Look-Through Cache memorija [Izvor: Autor]

## UPIS U KEŠ MEMORIJU

*Najveći problem kod upisa u keš memoriju jeste konzistentnost podataka u odnosu na iste podatke iz RAM memorije.*

Kod upisa u keš memoriju (L1 ili L2/L3) javlja se problem konzistentnosti podataka.

Podatak upisan u keš memoriju na određenom nivou ne odgovara njegovoj slici na drugom nivou.

Npr. podatak upisan u L1 keš memoriju nije isti onom u L2 keš memoriji, ni onome u operativnoj memoriji.

Kako bi se rešila konzistentnost podataka u memoriji na svim nivoima koriste se sledeće metode:

- Upis kroz (en. *write through*),
- Upis natrag (en. *write back*).

## UPIS NAZAD I UPIS-KROZ

*Samo L1 keš memorija ima najnoviju kopiju podatka dok su kopije na ostalim nivoima memorije zastarele.*

Kod upisa u keš memoriju (L1 ili L2/L3) javlja se problem konzistentnosti podataka.

Kada procesor upisuje rezultat operacije natrag u memoriju izvodi to na sledeći način.

Ako je blok prisutan u L1 keš memoriji, podatak se tamo ažurira i ujedno se ažurira i u L2 memoriji i u operativnoj memoriji. Ovakav pristup osigurava da su podaci na svim memorijskim nivoima u suglasnosti.

Ukoliko podatak nije u L1 keš memoriji, ne prebacuje se memorijski blok u L1 keš memoriju, nego se upisuje samo u L2 keš memoriju (ako je u njoj podatak prisutan) i u operativnu memoriju.

Kod **write back** pristupa, prilikom upisa podatka u memoriju proverava se prvenstveno da li je podatak u L1 keš memoriji pa ako jeste on se ažurira.

Njegova kopija na ostalim memorijskim nivoima ostaje neažurirana.

Tako samo L1 keš memorija ima najnoviju kopiju podatka dok su kopije na ostalim nivoima memorije zastarele.

Ukoliko lokacija u koju se upisuje nije u L1 keš memoriji moguća su sledeća rešenja:

- Podatak se upisuje samo u sledeći memorijsku nivo u kojoj je ta lokacija prisutna (L2 keš memorija ili operativna memorija).
- Učitava se linija u L1 keš memoriju i nakon njenog prebacivanja izvodi se promena sadržaja adresirane memorijske lokacije i linija se označava kao promenjena.

## STRATEGIJE ZAMENE LINIJA

*Najjednostavniji pristup zamene linija je slučajan odabir linije koja će se zameniti.*

Do sada je razmatran samo slučaj kada se unosi linija kada postoji slobodno mesta u keš memoriji.

Vremenom se keš memorija popuni i kod promašaja je potrebno isprazniti određenu liniju kako bi se oslobodilo mesto potrebnim podacima.

Potrebno je odrediti algoritam ili strategiju koja će se linija odabrati za zamenu, odnosno za izbacivanje iz brze memorije.

***Najjednostavniji pristup je slučajan odabir linije koja će se zameniti.***

Nepovoljnost kod ovog pristupa jeste ta da se zameni linija koja će se u skoroj budućnosti koristiti.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## KEŠ MEMORIJA I VIŠE JEZGARA

*L1 keš se uvek nalazi na čipu CPU-a.*

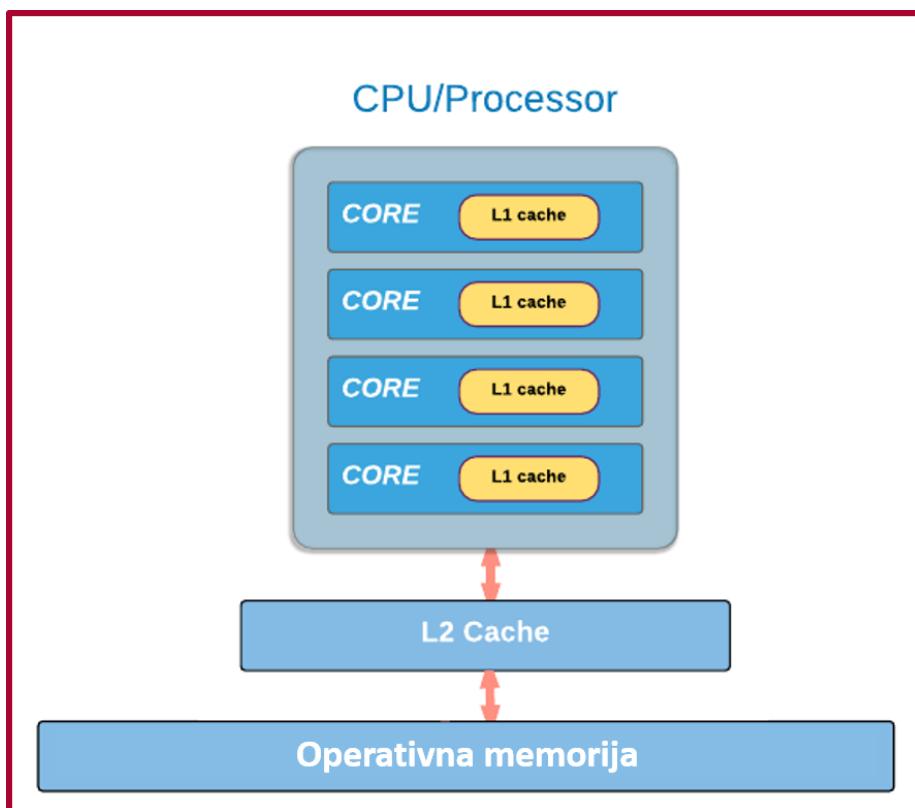
L1 keš se uvek nalazi na čipu CPU-a.

Kada procesor ima više jezgara, svako jezgro ima svoju L1 keš memoriju.

U zavisnosti od konstrukcije procesora, L2 keš memorija može biti zajednička za sva jezgra, a može biti i pojedinačna.

Noviji procesori imaju pojedinačne L2 keš memorije.

L3 keš memorije je (uglavnom za sad) zajednička za sva jezgra procesora.



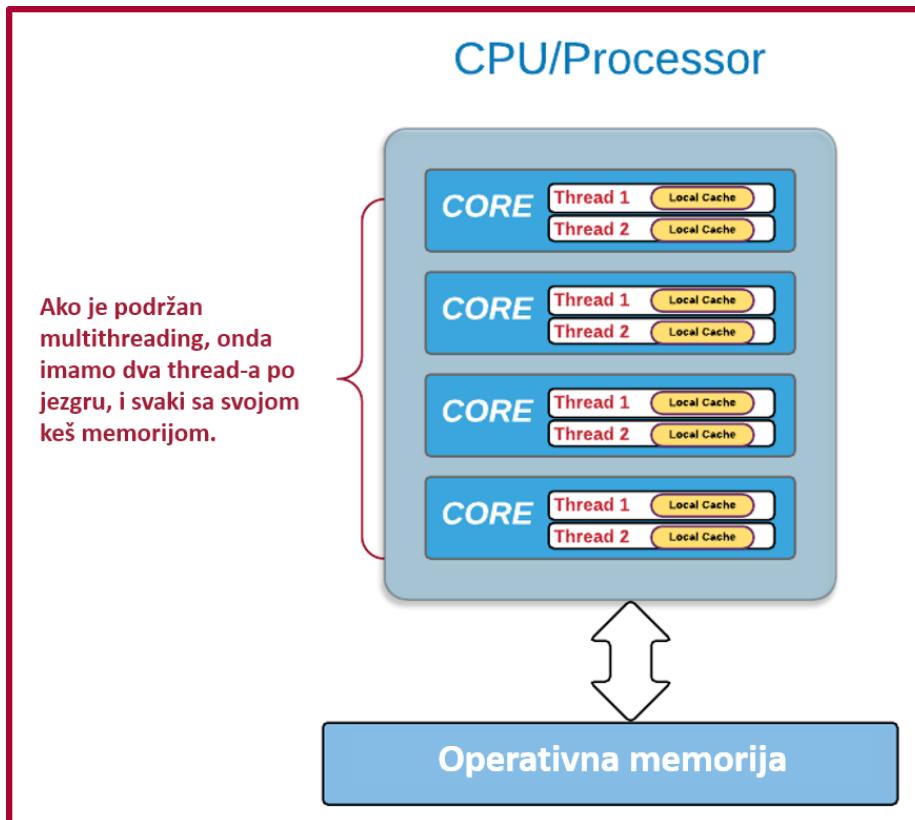
Slika 5.4 L1 i L2 keš memorija [Izvor: Autor].

## KEŠ MEMORIJA PO NITI PROCESORSKOG JEZGRA

*Moguće je imati lokalnu L1 (nekad i L2) keš memoriju po niti ukoliko procesor to podržava.*

Ukoliko procesor ima više logičkih jezgara po fizičkom jezgru (en. [hyperthreading](#)), onda postoji lokalna keš memorija za svaki thread.

Ukoliko procesor ne podržava hyperthreading, onda je L1 keš memorija po fizičkom jezgru.



Slika 5.5 Lokalna L1 keš memorija po niti. [Izvor: Autor]

## ▼ Poglavlje 6

### Pokazne vežbe

#### ZADATAK #1

*Prvi zadatak okvirno se radi 20 minuta*

##### **Zadatak #1 (20 minuta)**

Računar sa Intel i486DX procesorom poseduje 16 MB operativne memorije. Keš memorija procesora iznosi 8 kB, a preslikavanje je direktno.

Veličina bloka u memoriji je 32 B.

Prikazati organizaciju operativne i keš memorije, kao i polja fizičke adrese.

##### **Rešenje:**

Iz kapaciteta RAM memorije možemo pronaći dužinu fizičke adrese u bitovima:

$$A = \log_2 (C_{RAM}) = \log_2 (16 \text{ MB}) = \log_2 (2^4 \times 2^{20}) = \log_2 (2^{24}) = 24$$

Dužinu reči (ofseta) dobijamo iz veličine bloka:

$$B_{size} = \log_2 (32) = \log_2 (2^5) = 5$$

Ovoliko LSB bitova odvajamo iz fizičke adrese za offset (crveni bitovi na slici).

Broj blokova u operativnoj memoriji dobijamo kao količnik kapaciteta operativne memorije i veličine bloka:

$$N_{MB} = \frac{C_{RAM}}{B_{size}} = \frac{16 \text{ MB}}{32} = \frac{2^{24}}{2^5} = 2^{19}$$

Prvi blok je B(0), a poslednji blok je B(2<sup>19</sup>-1).

Broj blokova (linija) u keš memoriji dobijamo kao količnik kapaciteta keš memorije i veličine bloka:

$$N_{CB} = \frac{C_{Cache}}{B_{size}} = \frac{8 \text{ kB}}{32} = \frac{2^3 \times 2^{10}}{2^5} = 2^8 = 256$$

Prva linija je L(0), a poslednja linija je L(255).

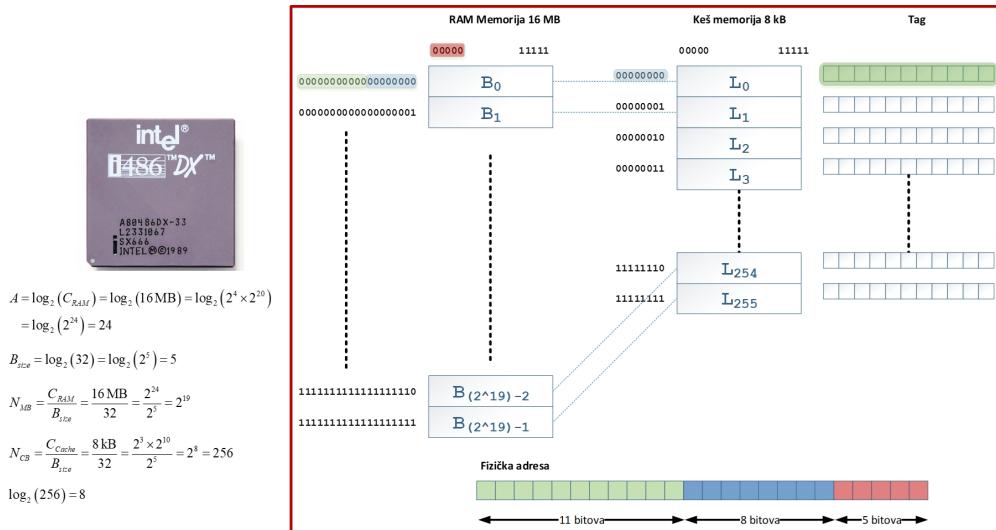
Za mapiranje 256 linija potrebno je 8 bitova, jer je

$$\log_2 (256) = 8$$

Ovoliko LSB bitova (posle ofseta) odvajamo za skup (plavi bitovi na slici). Konačno, preostale bitove iz fizičke adrese odvajamo za Tag bitove (zeleni bitovi na slici).

## REŠENJE PRVOG ZADATKA

### *Organizacija memorije za zadatku #1*



Slika 6.1 Rešenje zadatka #1 [Izvor: Autor]

## ZADATAK #2

### *Drugi zadatak okvirno se radi 20 minuta*

#### **Zadatak #2 (20 minuta)**

Računar sa AMD Athlon XP 2000+ procesorom poseduje 512 MB operativne memorije. Keš memorija procesora iznosi 2 x 64 kB (64 kB za kod, 64 za podatke), a preslikavanje je direktno.

Veličina bloka u memoriji je 64 B.

Prikazati organizaciju operativne i keš memorije, kao i polja fizičke adrese.

Tretirati ceo keš kao jedinstven.

#### **Rešenje:**

Iz kapaciteta RAM memorije možemo pronaći dužinu fizičke adrese u bitovima:

$$A = \log_2(C_{RAM}) = \log_2(512\text{ MB}) = \log_2(2^9 \times 2^{20}) = \log_2(2^{29}) = 29$$

Dužinu reči (ofseta) dobijamo iz veličine bloka:

$$B_{size} = \log_2(64) = \log_2(2^6) = 6$$

Ovoliko LSB bitova odvajamo iz fizičke adrese za ofset (crveni bitovi na slici). Broj blokova u operativnoj memoriji dobijamo kao količnik kapaciteta operativne memorije i veličine bloka:

$$N_{MB} = \frac{C_{RAM}}{B_{size}} = \frac{512 \text{ MB}}{64} = \frac{2^{29}}{2^6} = 2^{23}$$

Prvi blok je  $B(0)$ , a poslednji blok je  $B(2^{23}-1)$ .

Broj blokova (linija) u keš memoriji dobijamo kao količnik kapaciteta keš memorije i veličine bloka:

$$N_{CB} = \frac{C_{Cache}}{B_{size}} = \frac{128 \text{ kB}}{64} = \frac{2^7 \times 2^{10}}{2^6} = 2^{11} = 2048$$

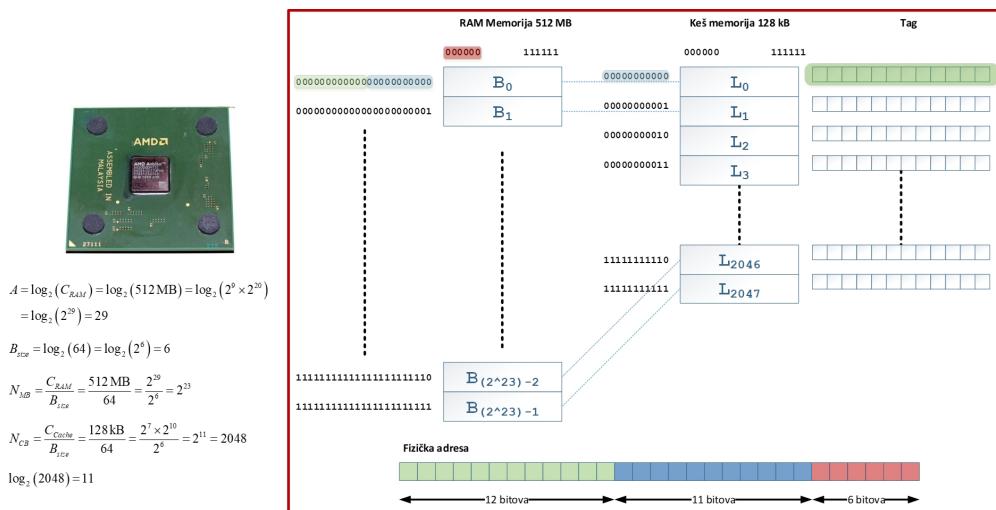
Prva linija je  $L(0)$ , a poslednja linija je  $L(2047)$ . Za mapiranje 2048 linija potrebno je 11 bitova, jer je

$$\log_2 (2048) = 11$$

Ovoliko LSB bitova (posle ofseta) odvajamo za skup (plavi bitovi na slici). Konačno, preostale bitove iz fizičke adrese odvajamo za Tag bitove (zeleni bitovi na slici).

## REŠENJE DRUGOG ZADATKA

### Organizacija memorije za zadatak #2



Slika 6.2 Rešenje zadatka #2 [Izvor: Autor]

## ZADATAK #3

*Treći zadatak okvirno se radi 20 minuta.*

### Zadatak #3 (20 minuta)

Računar sa Intel i5 2450M procesorom poseduje 4 GB operativne memorije.

Keš memorija procesora iznosi  $2 \times (2 \times 64 \text{ kB})$  (za svako jezgro po 64 kB za instrukcije, i po 64 za podatke), a preslikavanje je 8-struko set-asocijativno. Veličina bloka u memoriji je 64 B.

Prikazati organizaciju operativne i keš memorije, kao i polja fizičke adrese.  
Tretirati ceo keš kao jedinstven.

**Rešenje:**

Iz kapaciteta RAM memorije možemo pronaći dužinu fizičke adrese u bitovima:

$$A = \log_2 (C_{RAM}) = \log_2 (4 \text{ GB}) = \log_2 (2^2 \times 2^{30}) = \log_2 (2^{32}) = 32$$

Dužinu reči (ofseta) dobijamo iz veličine bloka:

$$B_{size} = \log_2 (64) = \log_2 (2^6) = 6$$

Ovoliko LSB bitova odvajamo iz fizičke adrese za offset (crveni bitovi na slici).

Broj blokova u operativnoj memoriji dobijamo kao količnik kapaciteta operativne memorije i veličine bloka:

$$N_{MB} = \frac{C_{RAM}}{B_{size}} = \frac{4 \text{ GB}}{64} = \frac{2^{32}}{2^6} = 2^{26}$$

Prvi blok je B(0), a poslednji blok je B(2<sup>26</sup>-1).

Broj linija u keš memoriji dobijamo kao količnik ukupnog kapaciteta keš memorije i veličine bloka:

$$N_{CB} = \frac{C_{Cache}}{B_{size}} = \frac{256 \text{ kB}}{64} = \frac{2^8 \times 2^{10}}{2^6} = 2^{12} = 4096$$

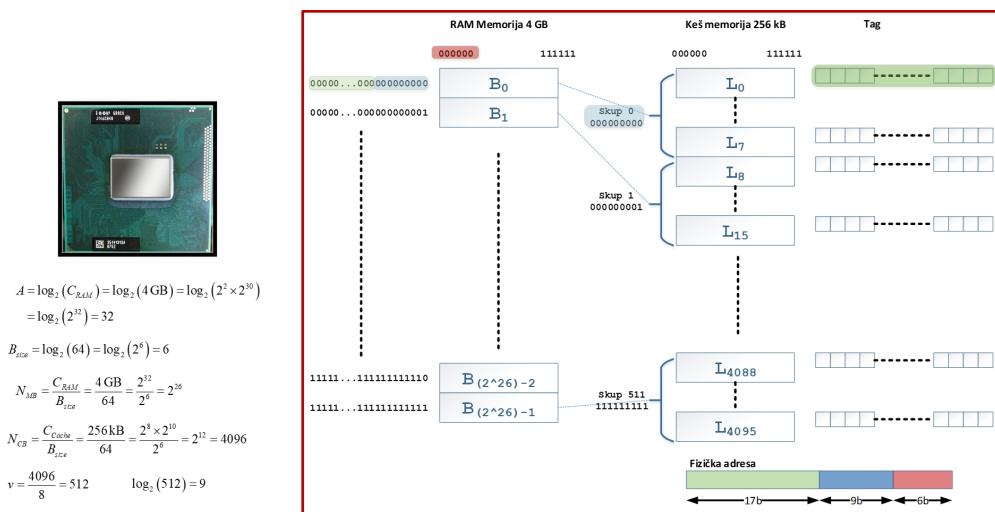
Prva linija je L(0), a poslednja linija je L(4095). Budući da je u pitanju 8-struko set-asocijativno preslikavanje skupova, broj skupova računamo kao količnik broja linija i broja linija po skupu.

$$v = \frac{4096}{8} = 512$$

Za 512 skupa potrebno je 9 bitova. Ovoliko LSB bitova (posle ofseta) odvajamo za skup (plavi bitovi na slici). Konačno, preostale bitove iz fizičke adrese odvajamo za Tag bitove (zeleni bitovi na slici).

## REŠENJE TREĆEG ZADATKA

*Organizacija memorije za zadatak #3*



Slika 6.3 Rešenje zadatka #3 [Izvor: Autor]

## ZADATAK #4

*Četvrti zadatak okvirno se radi 30 minuta.*

### Zadatak #4 (30 minuta)

Telefon Apple iPhone X ima processor Apple A11 i poseduje 2 GB operativne memorije. Keš memorija procesora iznosi  $2 \times 64\text{ kB}$  ( $64\text{ kB}$  za instrukcije,  $64\text{ kB}$  za podatke), a preslikavanje je direktno.

Veličina bloka u memoriji je 32 B.

Prikazati organizaciju operativne i keš memorije, kao i polja fizičke adrese.

Tretirati ceo keš kao jedinstven.

Rešenje:

Iz kapaciteta RAM memorije možemo pronaći dužinu fizičke adrese u bitovima:

$$A = \log_2(C_{RAM}) = \log_2(2\text{ GB}) = \log_2(2^1 \times 2^{30}) = \log_2(2^{31}) = 31$$

Dužinu reči (offseta) dobijamo iz veličine bloka:

$$B_{size} = \log_2(32) = \log_2(2^5) = 5$$

Ovoliko LSB bitova odvajamo iz fizičke adrese za offset (crveni bitovi na slici).

Broj blokova u operativnoj memoriji dobijamo kao količnik kapaciteta operativne memorije i veličine bloka:

$$N_{MB} = \frac{C_{RAM}}{B_{size}} = \frac{2\text{ GB}}{32} = \frac{2^{31}}{2^5} = 2^{26}$$

Prvi blok je  $B(0)$ , a poslednji blok je  $B(2^{26}-1)$ .

Broj blokova (linija) u keš memoriji dobijamo kao količnik ukupnog kapaciteta keš memorije i veličine bloka:

$$N_{CB} = \frac{C_{Cache}}{B_{size}} = \frac{128 \text{ kB}}{32} = \frac{2^7 \times 2^{10}}{2^5} = 2^{12} = 4096$$

Prva linija je L(0), a poslednja linija je L(4095).

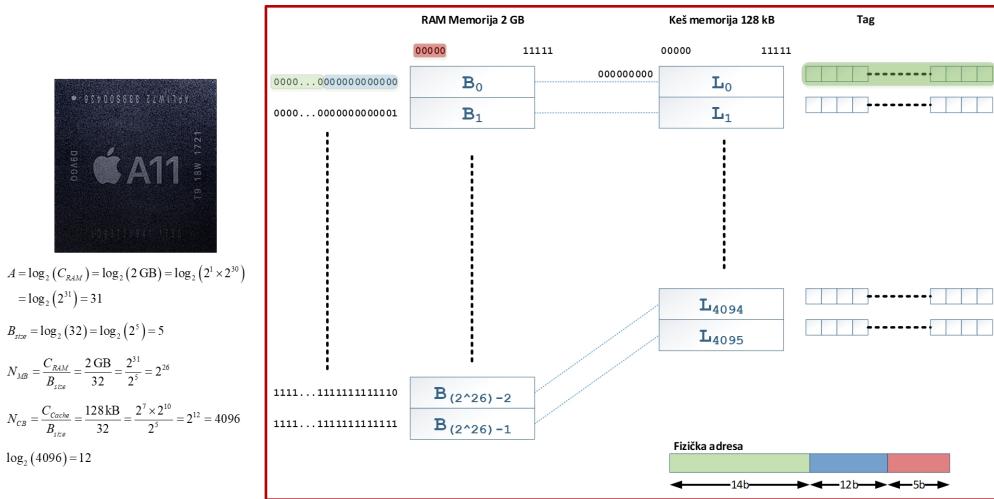
Za mapiranje 4096 linija potrebno je 12 bitova, jer je:

$$\log_2 (4096) = 12$$

Ovoliko LSB bitova (posle ofseta) odvajamo za skup (plavi bitovi na slici). Konačno, preostale bitove iz fizičke adrese odvajamo za Tag bitove (zeleni bitovi na slici).

## REŠENJE ČETVRTOG ZADATKA

### Organizacija memorije za zadatak #4



Slika 6.4 Rešenje zadatka #4 [Izvor: Autor]

## ✓ Poglavlje 7

### Zadaci za samostalni rad

#### ZADACI ZA SAMOSTALNI RAD STUDENATA

*Zadaci za samostalni rad rade se okvirno 20 minuta po zadatku.*

##### **Zadatak #1**

Pronaći specifikacije procesora na Vašem računaru. Pronaći kapacitet keš memorije i tip preslikavanja (ukoliko ne piše tip preslikavanja, tretirati da je preslikavanje direktno).

Na osnovu kapaciteta RAM memorije računara, uraditi mapiranje blokova RAM memorije u linije keš memorije.

*Napomena:* Ukoliko RAM memorija nije stepen broja 2, smanjiti kapacitet na prvi stepen broja 2

Primer: 6GB smanjiti na 4GB.

##### **Zadatak #2**

Pronaći specifikacije procesora na Vašem pametnom telefonu. Pronaći kapacitet keš memorije i tip preslikavanja (ukoliko ne piše tip preslikavanja, tretirati da je preslikavanje direktno).

Na osnovu kapaciteta RAM memorije telefona, uraditi mapiranje blokova RAM memorije u linije keš memorije.

## ✓ Poglavlje 8

### Domaći zadatak

#### DOMAĆI ZADATAK #6

*Domaći zadatak #4 okvirno se radi 30 min*

**Domaći zadatak:**

Računar sa AMD Duron 750 procesorom poseduje 128 MB operativne memorije.

Keš memorija procesora iznosi 64 kB, a preslikavanje je dvostruko set-asocijativno (2-way set associative).

Veličina bloka u memoriji je 128 B.

Prikazati organizaciju operativne i keš memorije, kao i polja fizičke adrese.

Informacije o procesoru možete naći na:

<https://www.cpu-world.com/CPUs/K7/AMD-Duron%20750%20-%20D750AUT1B.html>

**Predaja domaćeg zadatka:**

Domaći zadatak slati odgovarajućem predmetnom asistentu, sa predmetnim profesorom u CC.

Predati domaći zadatak koristeći .doc/docx uputstvo dato u prvoj lekciji.

## ✓ Poglavlje 9

### Zaključak

## ZAKLJUČAK

### *Rezime lekcije #6*

U ovoj lekciji bilo je reči o razlozima uvođenja keš memorije u hijerarhijski sistem memorija u računaru.

Opisane su funkcije mapiranja, uz primere za direktno, asocijativno i set-asocijativno mapiranu keš memoriju.

Nakon toga dat je osvrt na višnivoovske keš memorije koje koriste savremeni procesori.

U zadacima prikazan je postupak računanja broja blokova u RAM memoriji i broj linija u keš memoriji kao i način računanja fizičke adrese koju procesor može tražiti.

### **Literatura:**

A. Tanenbaum, Structured Computer Organization, Chapter 04, pp. 303 - 323.

