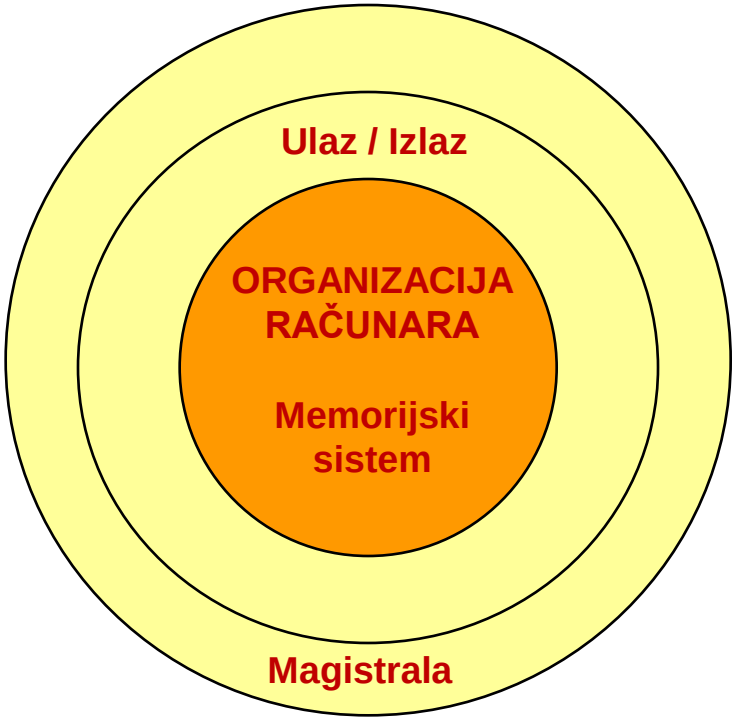
Memorijski sistem



TEME

- ✓ Memorijska hijerarhija
- ✓ Keš memorija
  - ✓ Tehnike preslikavanja
  - ✓ Tehnike zamene
  - ✓ Tehnike ažuriranja

Memorijska hijerarhija

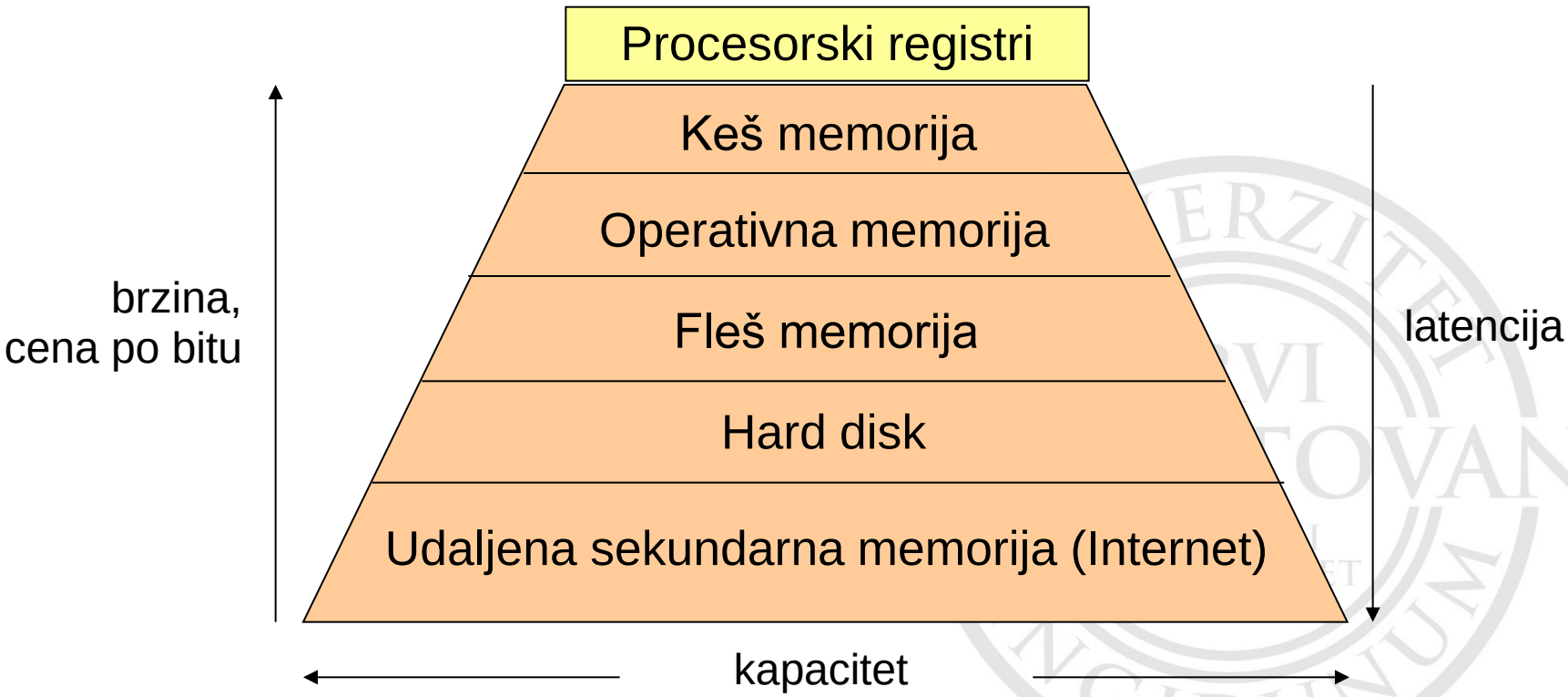
- Ideja o hijerarhijskoj organizaciji računarske memorije potiče još od *Von Neumann*-a (1946).

Memorijska hijerarhija se karakteriše parametrima:

- vreme pristupa (operacija upisa/čitanja)
- kapacitet (meri se u bajtovima)
- vreme ciklusa (vreme proteklo između dve uzastopne operacije upisa)
- latencija (vreme proteklo od zahteva za informacijom do pristupa prvom bitu te informacije)
- cena (valuta/megabajtu)

Memorijska hijerarhija

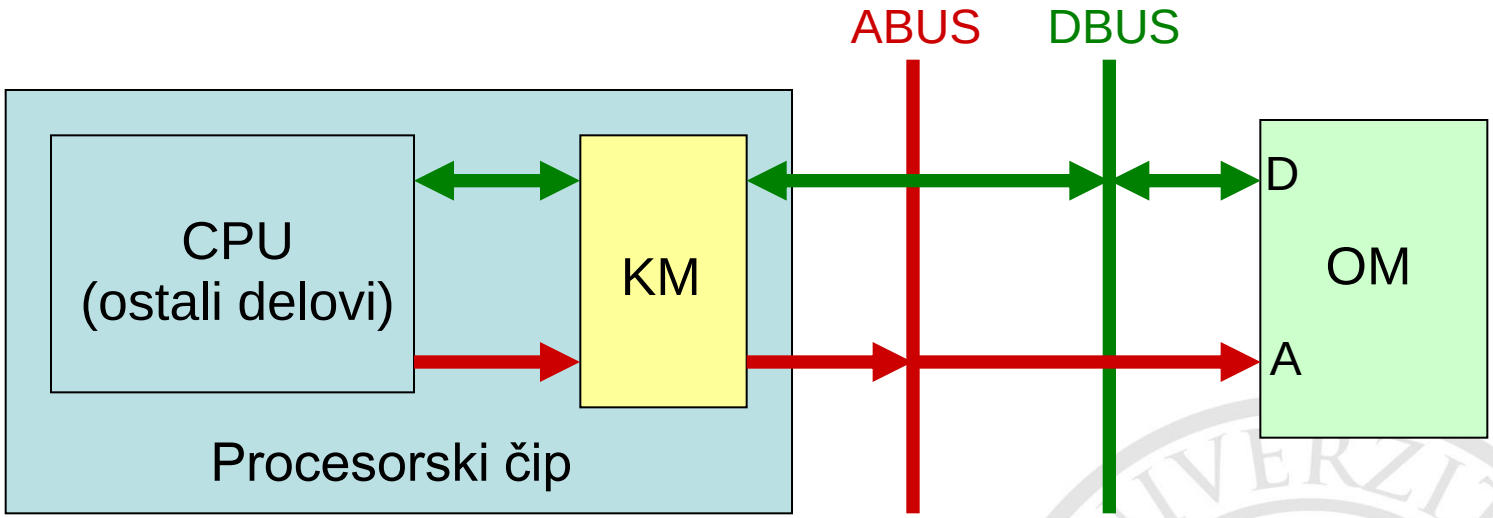
- Hijerarhija polazi od malih, skupih i relativno brzih memorijskih jedinica ka velikim, jeftinijim i relativno sporim jedinicama.



Keš memorija

- Keš memorija – KM rešava problem vremena pristupa tako što omogućava brži pristup sadržaju memorijskih lokacija.
- Realizuje se unutar procesora (ranije je bila posebna komponenta).
- Vreme pristupa KM je mnogo manje od vremena pristupa OM i vrlo blisko vremenu prenosa između registara.
- Cena po bitu KM je znatno veća nego kod OM, pa je kapacitet KM znatno manji od kapaciteta OM.

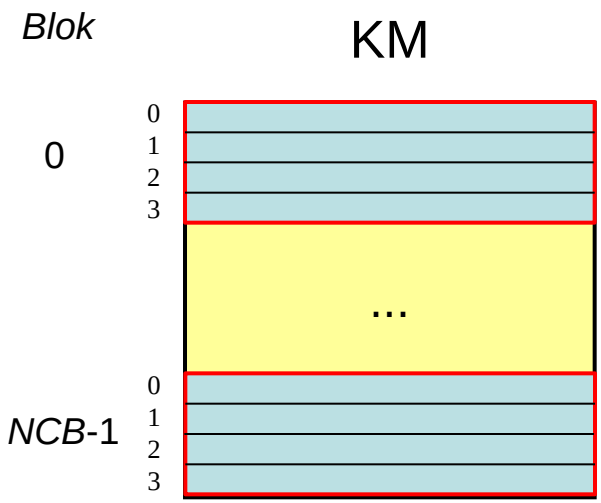
Princip rada

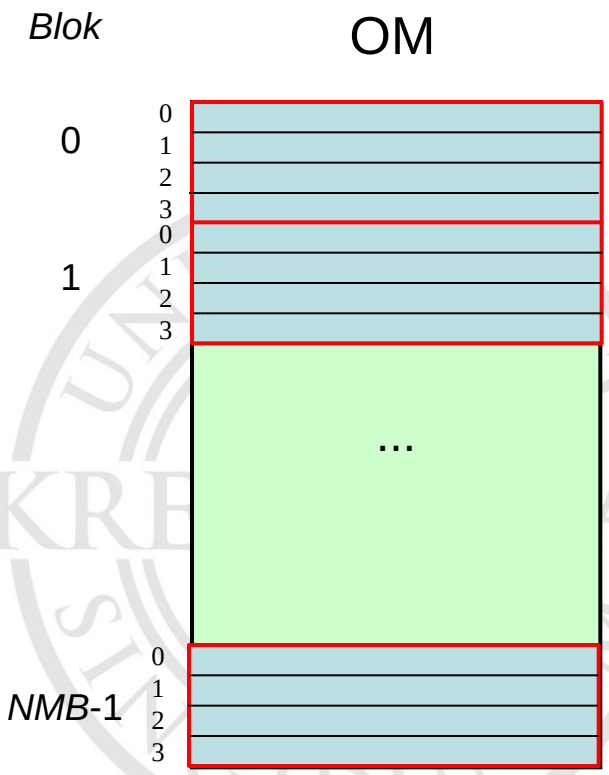


- CPU generiše adresu u OM
- proverava se da li je sadržaj sa te adrese u KM
- u početku nije, pa se sadržaj na toj i susednim adresama prenosi iz OM u KM
- pristupa se sadržaju u KM
- postupak se ponavlja za nekoliko sledećih adresa,
  tako da se KM postepeno puni
- u nekom trenutku, traženi sadržaj će biti u KM, pa će pristup biti brži

Blokovi

- Blok predstavlja skup uzastopnih memorijskih lokacija.
- OM i KM su podeljene u blokove iste veličine.

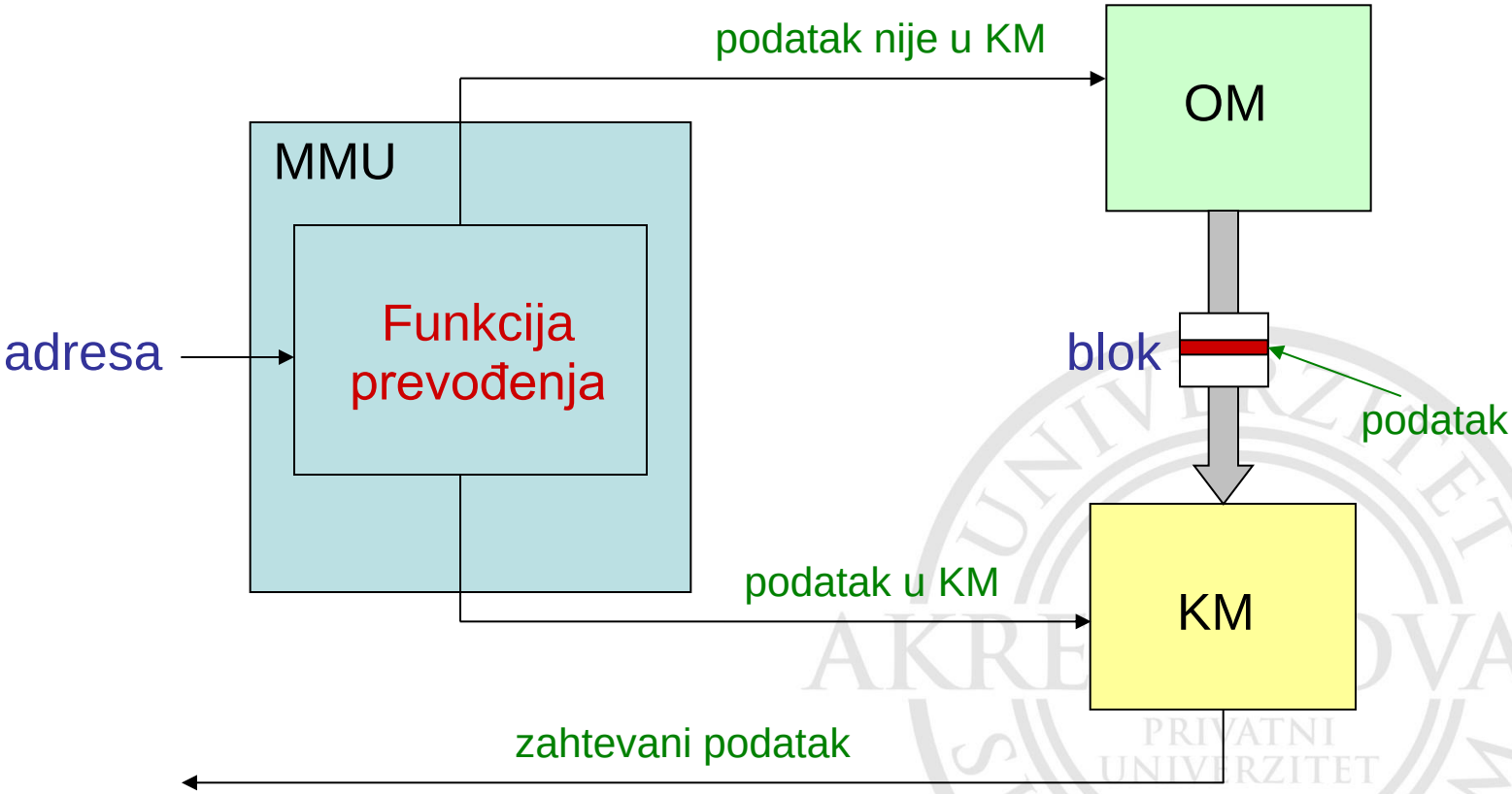




Mapiranje adresa

- Procesor generiše adresu za pristup podatku (adresu u OM).
- Podatak može da se nalazi u KM ili u OM, očigledno na različitim adresama zato što KM i OM imaju različite kapacitete.
- Da bi se došlo do željenog podatka, potrebno je obezbediti prevođenje, tj. mapiranje adrese.
- Mapiranje adresa je funkcija koju izvršava jedinica za upravljanje memorijom - MMU (*Memory Management Unit*)
- MMU se obično implementira kao deo CPU.

Mapiranje adresa



Problemi

1. Evidencija o tome koji blokovi OM se trenutno nalaze u KM i gde su smešteni – ovaj problem rešavaju tehnike preslikavanja

2. Odlučivanje o tome koji blok treba izbaciti iz pune KM kako bi se stvorio prostor za upis novog bloka – ovaj problem rešavaju tehnike zamene

3. Ažuriranje sadržaja OM u slučaju kada je bilo upisa u blok KM koji se izbacuje, jer tada sadržaji na istoj adresi u KM i OM nisu isti – ovaj problem rešavaju tehnike ažuriranja

Tehnike preslikavanja

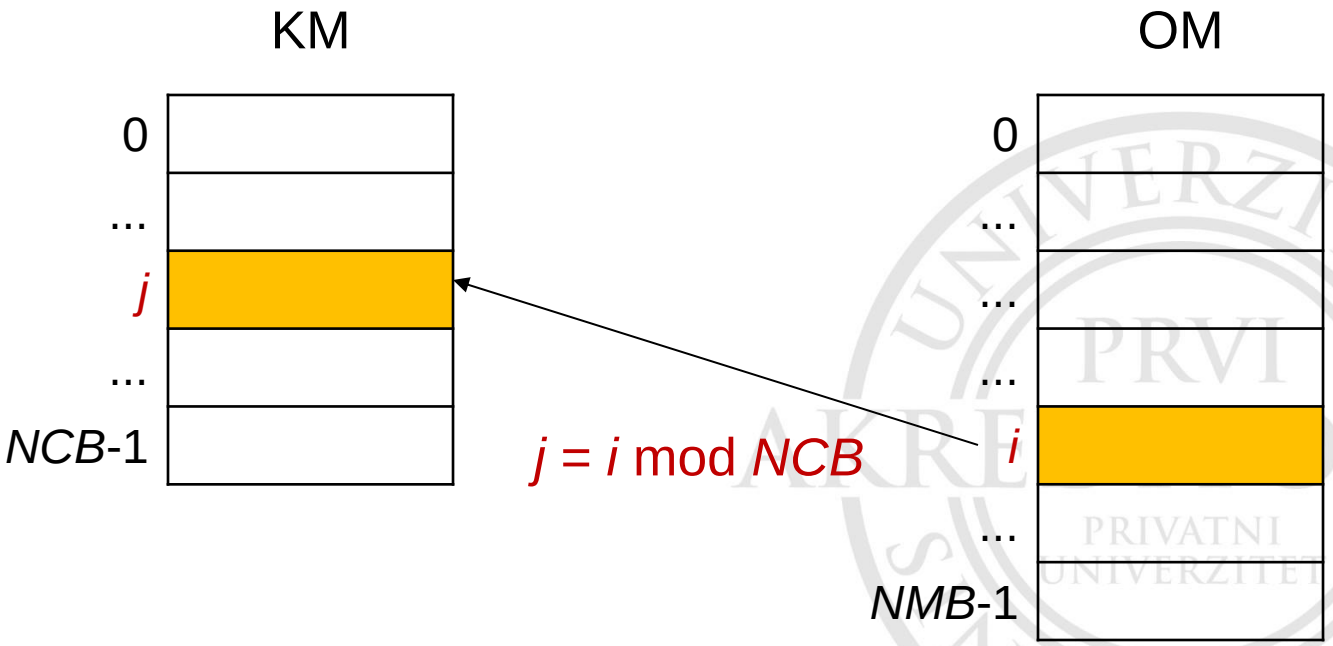
1) Direktno preslikavanje

2) Asocijativno preslikavanje

3) Set-asocijativno preslikavanje

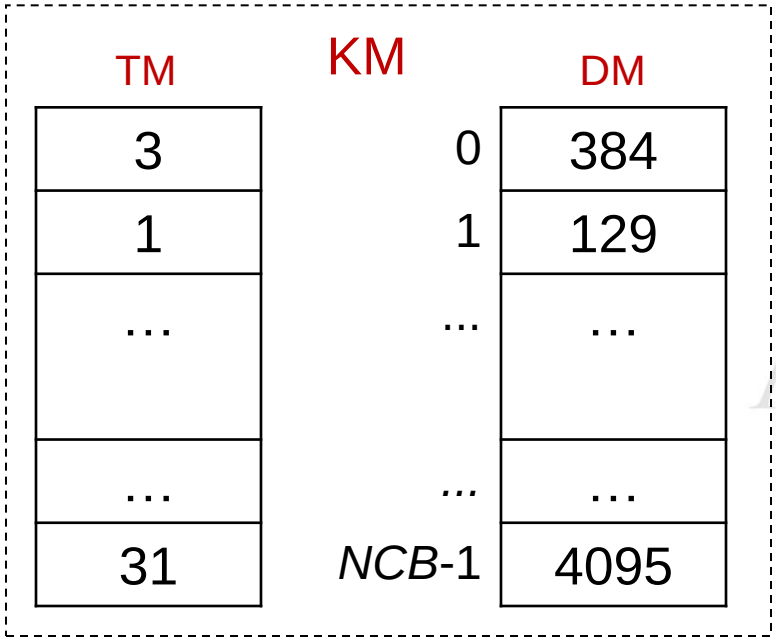
Direktno preslikavanje

- Najjednostavnija tehnika preslikavanja.
- Blok OM se smešta u fiksni blok KM.



Direktno preslikavanje

- Keš memorija se sastoji od TM (*Tag Memory*) i DM (*Data Memory*).
- U TM se čuva evidencija o tome koji se blokovi OM nalaze u KM.
- U DM se čuvaju blokovi podataka preneti iz OM u KM.



Direktno preslikavanje

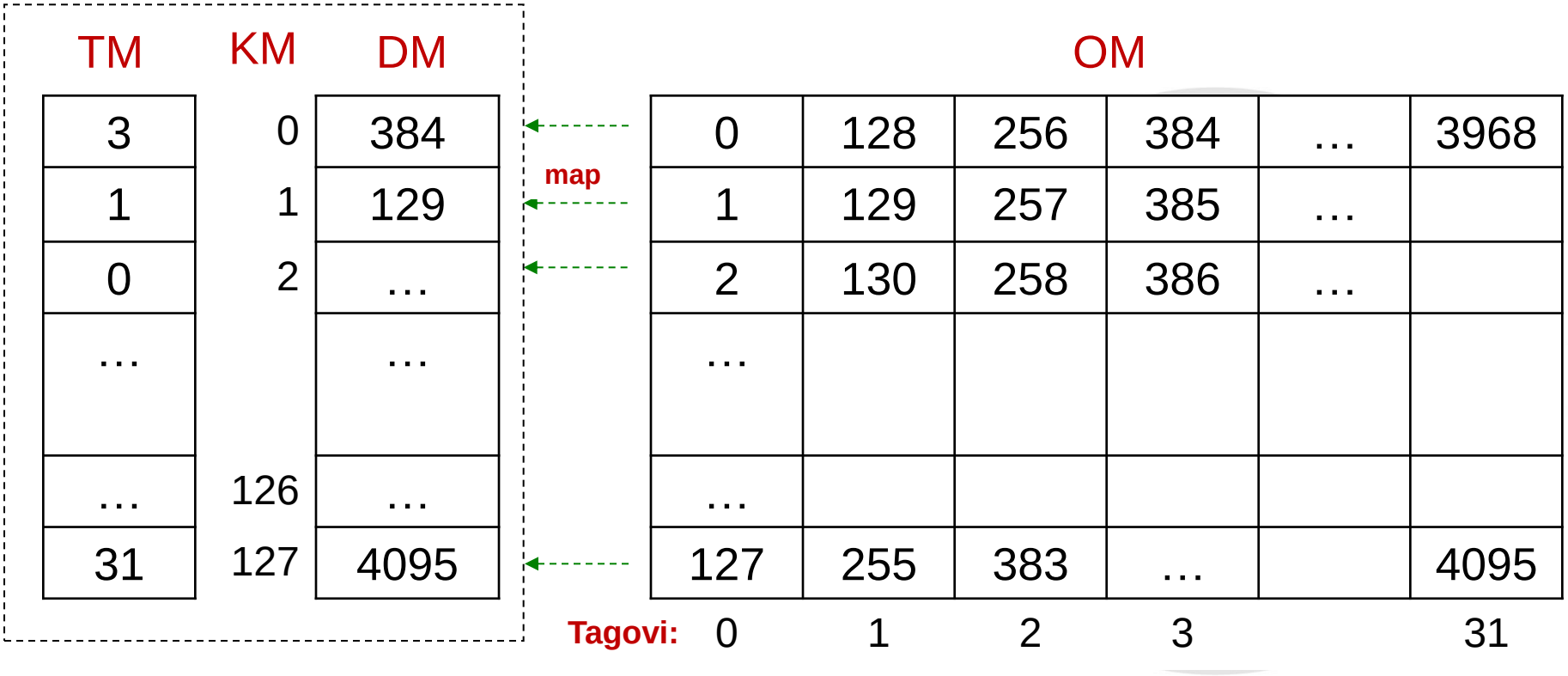
Primer 1

OM sadrži *NMB* = 4K blokova (4096 blokova)

KM sadrži *NCB* = 128 blokova

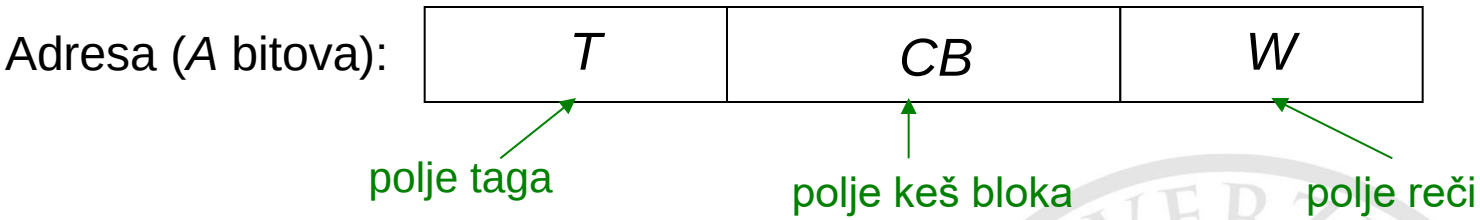
Veličina bloka je *B* = 16 memorijskih reči

Tag = *i* // *NCB* (*ceo deo*)



Direktno preslikavanje

MMU interpretira adresu dobijenu od procesora tako što je podeli u tri polja sa *T*, *CB* i *W* bitova.



Određivanje broja bitova u poljima

$$A = \log_2(B \cdot NMB)$$

$$T = \log_2(NMB/NCB) \qquad CB = \log_2 NCB \qquad W = \log_2 B$$

Direktno preslikavanje

Primer 2

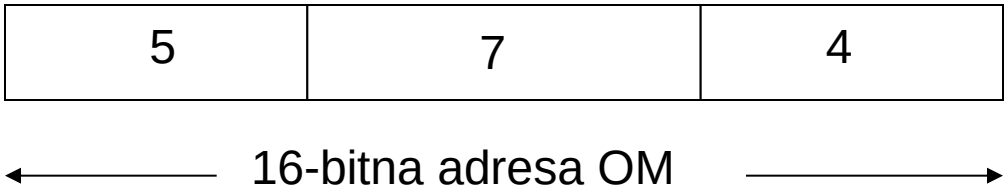
$NMB = 4096$ $NCB = 128$ $B = 16$

$A = \log_2(16 \cdot 4096) = 16$ bitova

$T = \log_2(NMB/NCB) = \log_2(4096/128) = 5$ bitova

$CB = \log_2 NCB = \log_2 128 = 7$ bitova

$W = \log_2 B = \log_2 16 = 4$ bita



Direktno preslikavanje

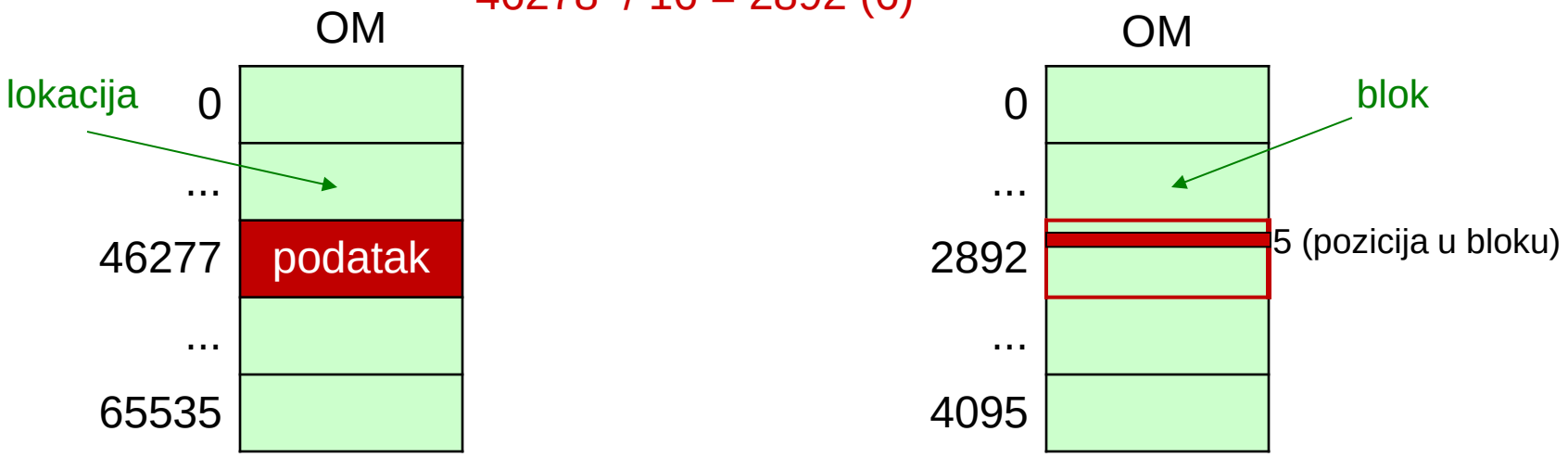
Primer 3

Pokazaćemo na primeru da je adresa koju generiše procesor istovremeno i adresa podatka u OM i adresa tog podatka u KM (ako se on nalazi u KM).

Neka je: *NMB* = 4096 *NCB* = 128 *B* = 16

Adresa koju je generisao procesor: $1011010011000101_{(2)} = 46277_{(10)}$

46278 / 16 = 2892 (6)



Napomena: Adrese, blokovi u OM i KM, pozicije reči unutar bloka, kao i tagovi, broje se počevši od 0!

Direktno preslikavanje

Primer 3 - nastavak

NMB = 4096 *NCB* = 128 *B* = 16

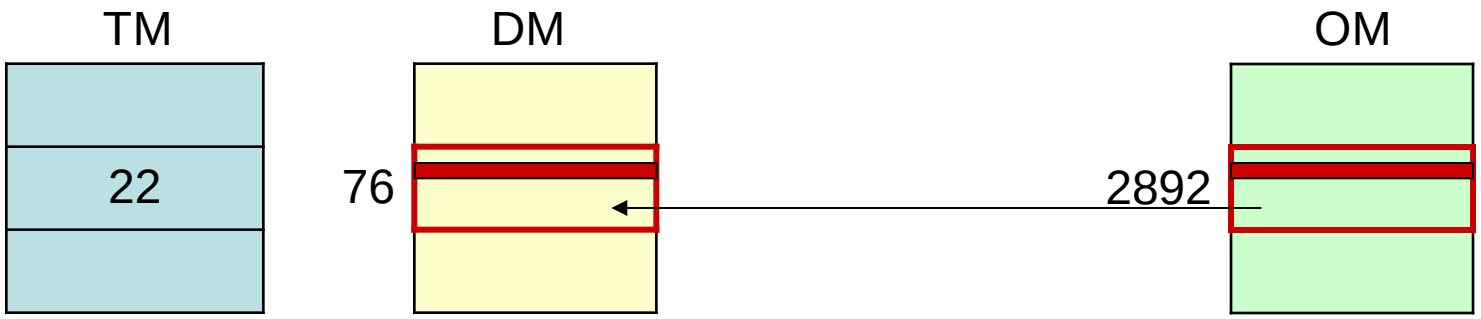
Adresa: $1011010011000101_{(2)} = 46277_{(10)}$ *i* = 2892 (blok u OM)

Računanje bloka KM po formuli: *j* = *i* mod *NCB* = 2892 mod 128 = 76

Računanje taga po formuli: Tag = *i* // *NCB* = 2892 // 128 = 22

Interpretacija MMU: 1011010011000101

T (5) :$10110_{(2)}=22_{(10)}$ *CB* (7) : $1001100_{(2)}=76_{(10)}$ W (4) : $0101_{(2)}=5_{(10)}$



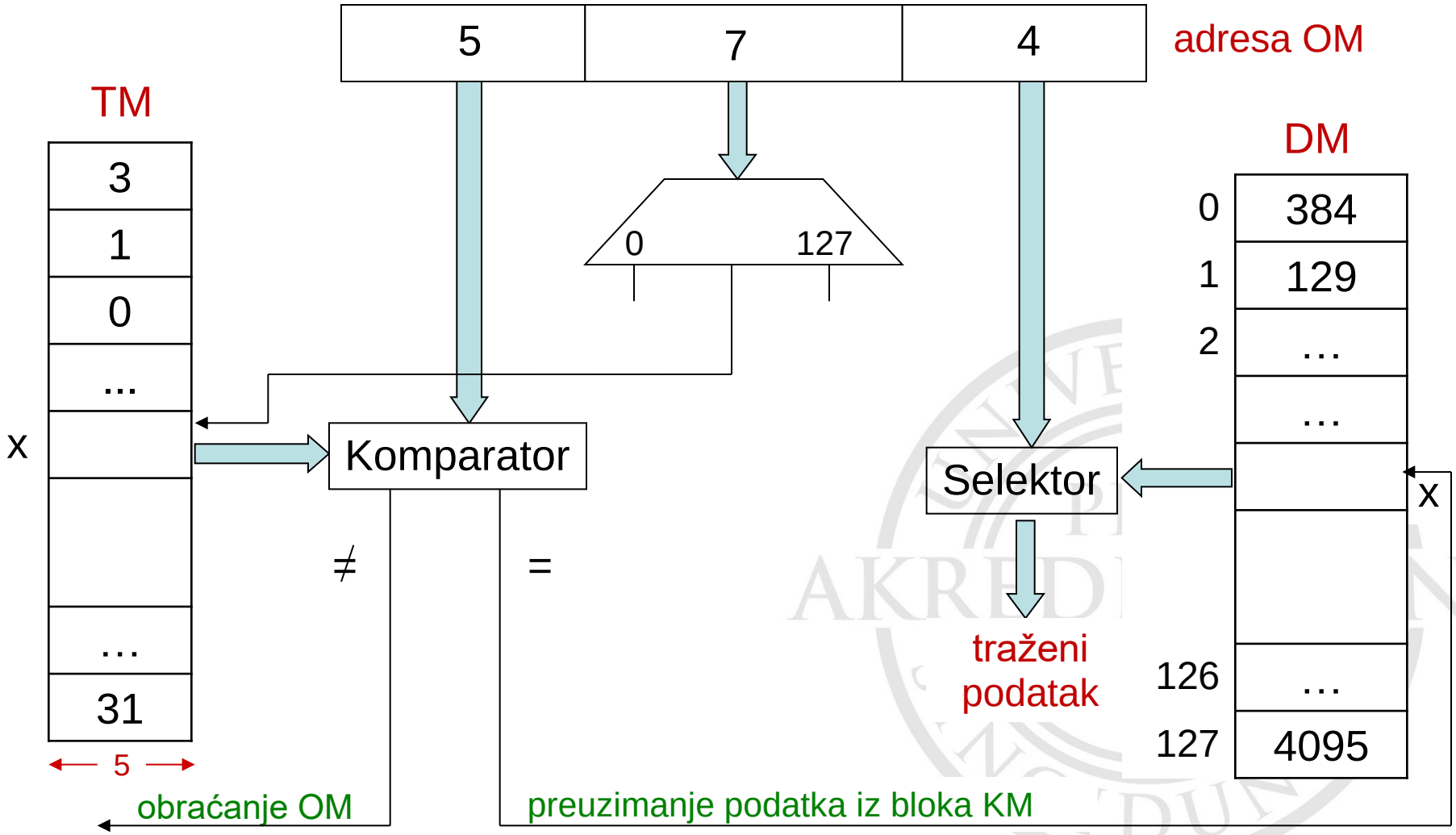
Direktno preslikavanje

T	CB	W

MMU primenjuje sledeći protokol:

1. Na osnovu polja *CB* određuje se keš blok u kome bi trebalo da se nalazi traženi podatak (ako je u KM).
2. Za taj blok se tag iz TM poredi sa vrednošću polja *T* u adresi i zaključuje da li je podatak u KM ili nije (iste vrednosti → podatak je u KM).
3. Ako je podatak u KM, njegova pozicija unutar bloka je određena vrednošću u polju *W.*
4. Ako podatak nije u KM, onda blok iz OM treba preneti u KM i podatak proslediti procesoru.

Direktno preslikavanje



Direktno preslikavanje

Direktno preslikavanje je više-na-jedan tehnika mapiranja.

Prednost:

- ✓ jednostavno, direktno određivanje mesta u KM gde će biti smešten blok OM

Nedostatak:

- ✓ neefikasno korišćenje KM (više memorijskih blokova konkurišu za isto mesto u KM, a možda postoje drugi, prazni keš blokovi)

Asocijativno preslikavanje

- Fleksibilnije od direktnog preslikavanja.
- Blok koji dolazi iz OM se smešta u bilo koji raspoloživ blok KM.

MMU interpretira adresu koju generiše procesor u vidu dva polja sa *T* i *W* bitova:

- *T* – broj bloka OM koji se nalazi u KM
- *W* – identifikuje podatak unutar bloka

$T = \log_2 NMB$ $W = \log_2 B$ $A = \log_2(B \cdot NMB) = T + W$

Asocijativno preslikavanje

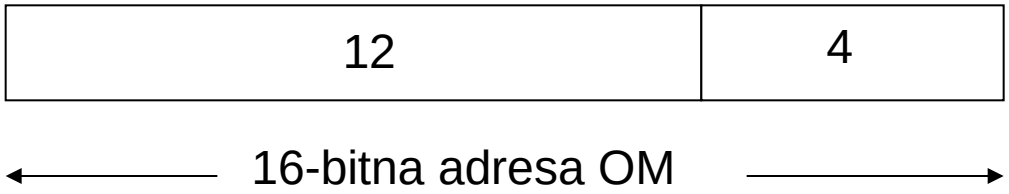
Primer 4

$NMB = 4096$ $NCB = 128$ $B = 16$

$A = \log_2(16 \cdot 4096) = 16$ bitova

$T = \log_2 NMB = \log_2 4096 = 12$ bitova

$W = \log_2 B = \log_2 16 = 4$ bita

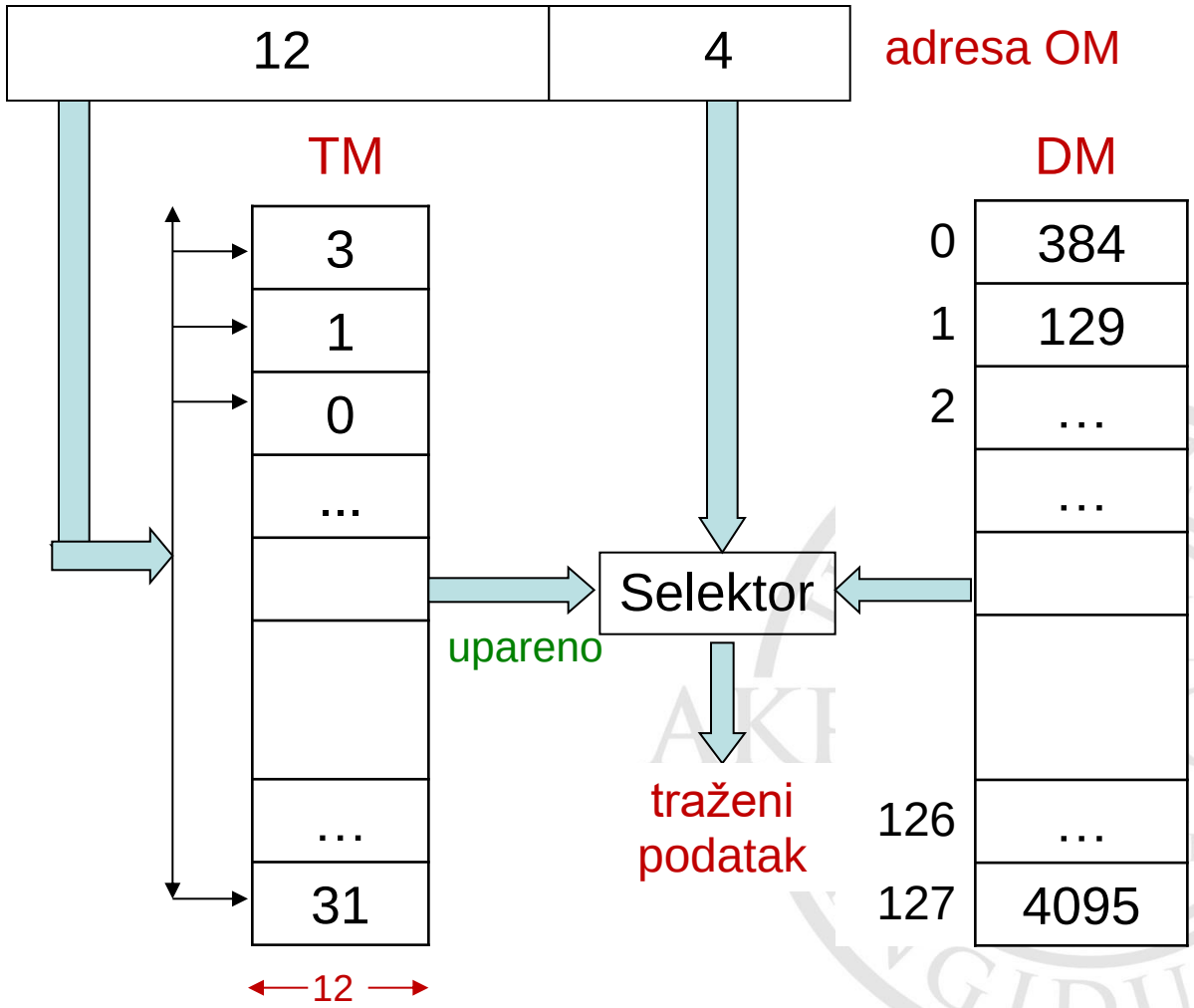


Asocijativno preslikavanje

Da bi se pristupilo podatku čiju je adresu generisao procesor, MMU primenjuje sledeći protokol:

1. Ispituje se da li se vrednost u polju *T* u adresi nalazi među postojećim tagovima u TM.
2. Ako se nalazi, traženi podatak je u odgovarajućem bloku KM.
3. U tom bloku KM, podatak se nalazi u lokaciji čija pozicija unutar bloka odgovara vrednosti u polju *W.*
4. Ako se *T* ne nalazi u TM, podatak nije u KM, pa blok iz OM treba preneti u KM i podatak proslediti procesoru.

Asocijativno preslikavanje



Asocijativno preslikavanje

Ako bi pretraživanje TM bilo sekvencijalno, trajalo bi dugo.

Zato se tagovi čuvaju u asocijativnoj memoriji (paralelno pretraživanje).

Prednost:

- ✓ nema restrikcija po pitanju mesta u KM gde će se smestiti blok koji dolazi iz OM

Nedostatak:

- ✓ potreban hardver za asocijativno pretraživanje memorije tagova

Set-asocijativno preslikavanje

- KM je podeljenja u setove, pri čemu svaki set sadrži isti, zadati broj blokova.
- Blok koji dolazi iz OM unosi se u set KM prema formuli:

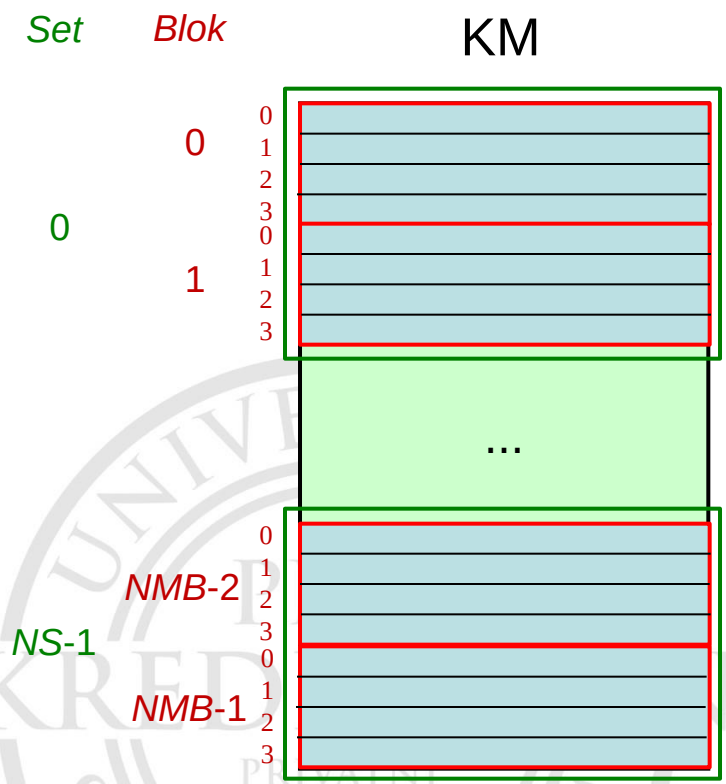
$$s = i \bmod NS$$

gde je

i – broj bloka OM

NS – broj setova u KM

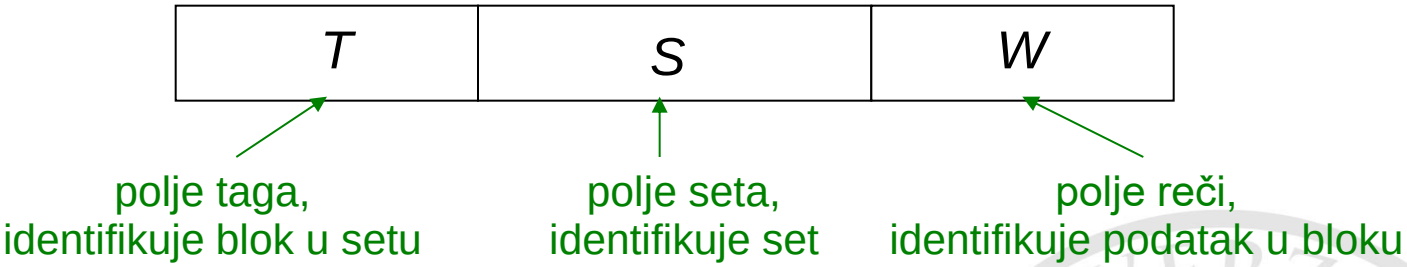
s – broj seta u KM u koji se unosi blok OM



- Blok *i* iz OM se unosi u bilo koji blok u setu *s* u KM.

Set-asocijativno preslikavanje

MMU interpretira adresu koju generiše procesor u vidu tri polja sa *T*, *S* i *W* bitova.



Određivanje broja bitova u poljima

NS - broj setova u KM *BS* - broj blokova u setu *B* - broj reči u bloku

$T = \log_2(NMB/NS)$ $S = \log_2 NS$ $W = \log_2 B$

$A = \log_2(B \cdot NMB)$

Set-asocijativno preslikavanje

Primer 5

$NMB = 4096$ $NCB = 128$ $B = 16$ $BS = 4$

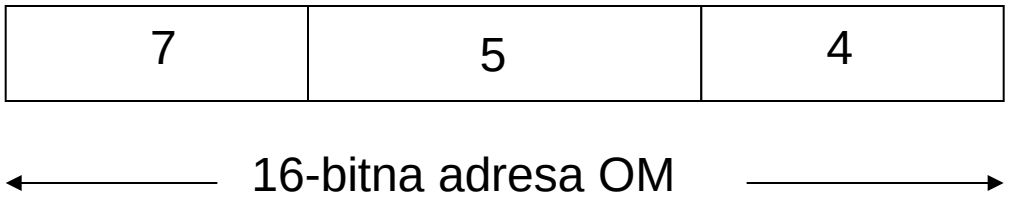
$NS = 128/4 = 32$

$A = \log_2(16 \cdot 4096) = 16$ bitova

$T = \log_2(NMB/NS) = \log_2(4096/32) = 7$ bitova

$S = \log_2 NS = \log_2 32 = 5$ bitova

$W = \log_2 B = \log_2 16 = 4$ bita

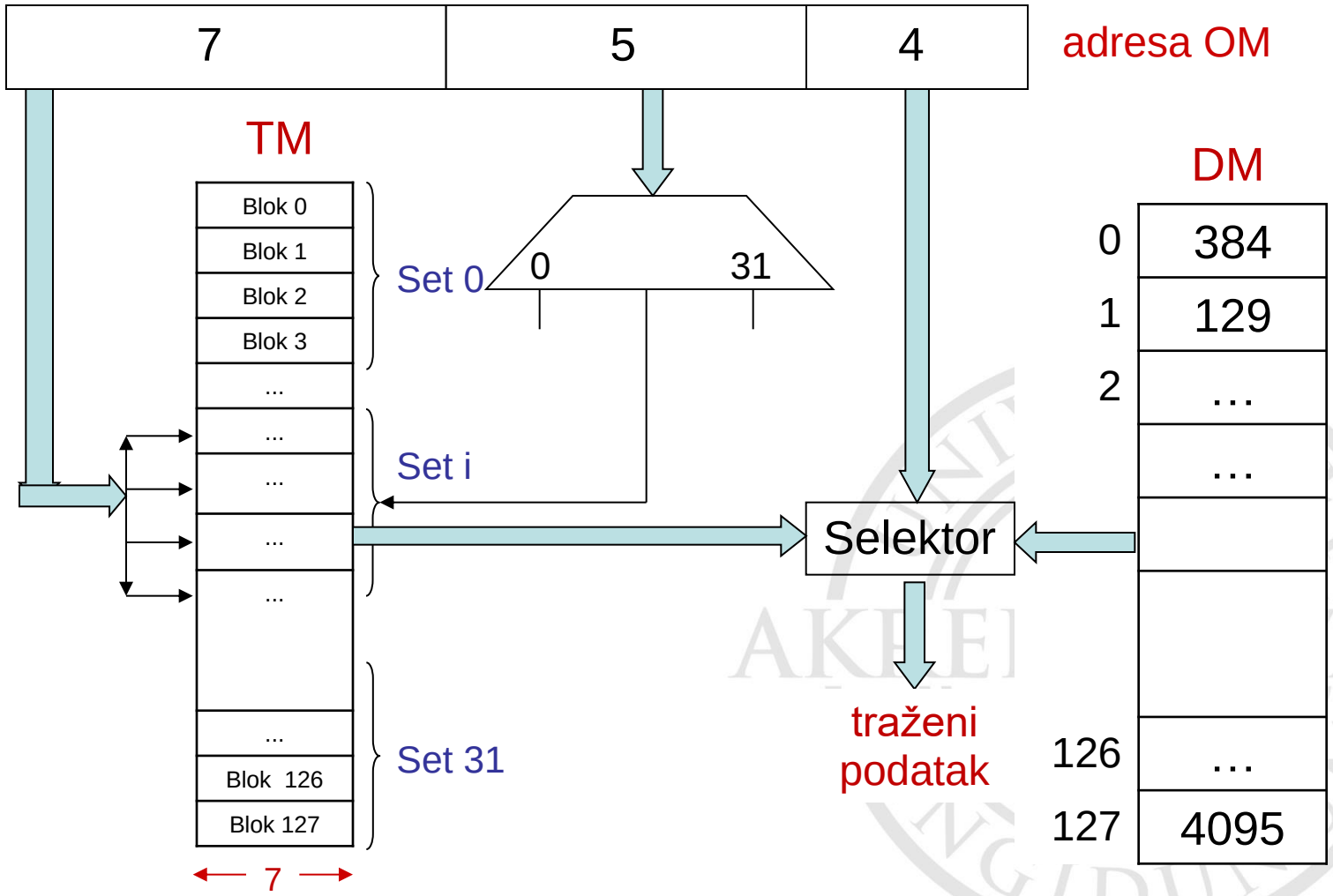


Set-asocijativno preslikavanje

Da bi se pristupilo podatku čiju je adresu generisao procesor, MMU primenjuje sledeći protokol:

1. Na osnovu polja *S* direktno se određuje set u koji se blok OM mapira.
2. Ispituje se da li se vrednost polja *T* u adresi nalazi među postojećim tagovima u TM za dati set.
3. Ako se nalazi, to znači da je taj blok već u KM, pa se podatak pronalazi unutar bloka pomoću vrednosti u polju *W.*
4. Ako se *T* ne nalazi u TM, blok iz OM treba preneti u odgovarajući set KM i podatak proslediti procesoru.

Set-asocijativno preslikavanje



Set-asocijativno preslikavanje

- Tagovi se čuvaju u asocijativnoj memoriji da bi pretraživanje bilo brže (paralelno pretraživanje).
- Po osobinama, ova tehnika je kombinacija direktnog i asocijativnog preslikavanja.
- Tehnika nije toliko efikasna kao asocijatvno preslikavanje, ali je preuzela jednostavnost direktnog preslikavanja.

Poređenje tehnika preslikavanja

Tehnika preslikavanja	Jednostavnost	Asocijativno pretraživanje tagova	Očekivano iskorišćenje keša	Tehnika zamene
Direktno	da	ne	nisko	nepotrebna
Asocijativno	ne	da	visoko	potrebna
Set-asocijativno	srednja	umereno	srednje	potrebna

Tehnike zamene

- Rešavaju problem izbora bloka u punoj KM koji će biti zamenjen blokom koji dolazi iz OM.

Izbor bloka može biti:

- po slučajnom izboru (*Random Selection*)
- po vremenu provedenom u KM (*FIFO: First-In-First-Out*)
- po trenutku poslednjeg korišćenja bloka (*LRU: Least Recently Used*)

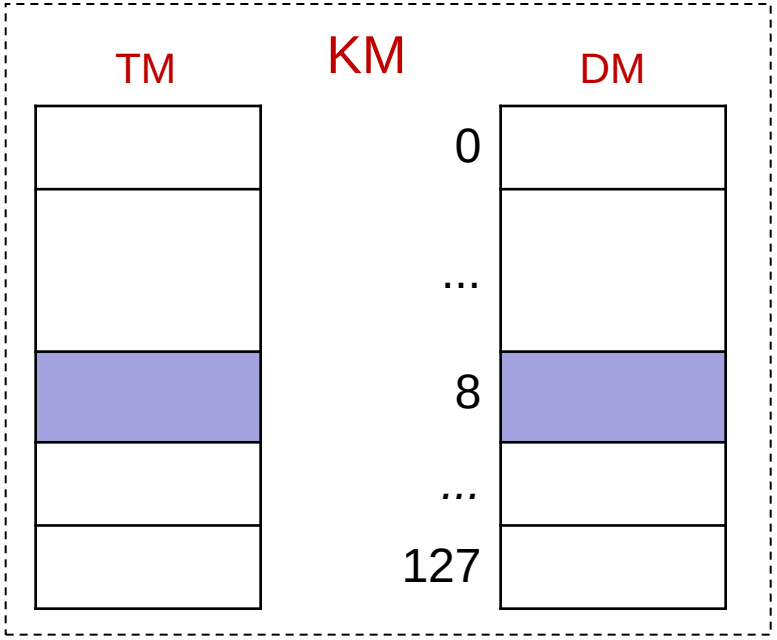
Slučajan izbor

- U računaru postoji generator slučajnih brojeva (GSB) koji po uključenju računara počinje da generiše brojeve iz opsega od 0 do *N*-1.
- Tehnika se zasniva na tome da se broj bloka *i* koga treba izbaciti iz KM određuje izlazom GSB u trenutku zamene *g*.
- Tehnika je vrlo jednostavna.

Primer 6

$NCB = 128$
GSB $\rightarrow g = 13320$

$i = g \bmod NCB = 13320 \bmod 128 = 8$



FIFO tehnika

- Zasniva na tome da se izbacuje blok koji je najviše vremena proveo u KM, bez obzira na njegovo korišćenje u tom periodu.
- Dakle, izbacuje se blok koji je od svih blokova koji su trenutno u KM najranije unet u nju.
- Tehnika zahteva vođenje evidencije o trenucima unosa blokova u KM, pa je složenija od slučajnog izbora.

LRU tehnika

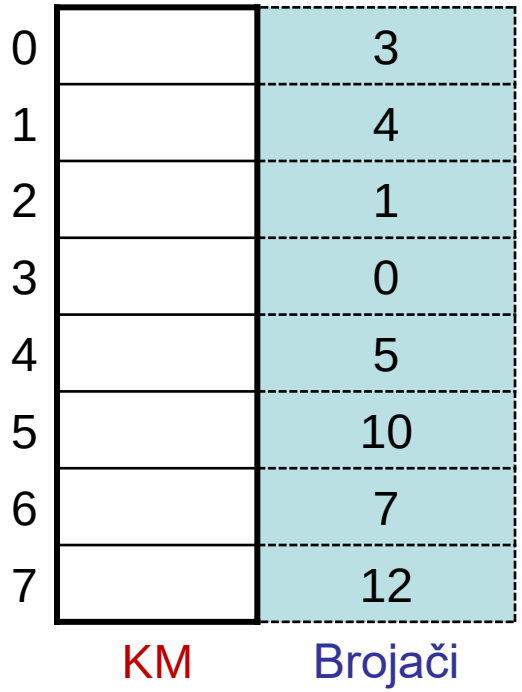
- Tehnika se zasniva na tome da se iz KM izbacuje blok koji je, od svih blokova trenutno prisutnih u KM, najranije poslednji put korišćen.
- Dakle, izbacuje se blok koji naduže nije bio korišćen (nije mu se pristupalo u cilju upisa/čitanja podatka iz njega).
- Najefikasnija tehnika, zato što prati istoriju upotrebe blokova.
- Tehnika zahteva korišćenje keš kontrolera koji čuva istoriju referenci na sve blokove u KM za sve vreme dok su tu.

LRU tehnika

Jedna implementacija keš kontrolera

- Svakom bloku u KM je pridružen brojač.
- Vrednost brojača je veća ako je blok ranije korišćen.
- Svi brojači imaju različite vrednosti, pa je poznat redosled korišćenja blokova.

Primer 7



Univerzitet Singidunum
www.singidunum.ac.rs

LRU tehnika

Primer 8

Procesor zahteva podatke iz blokova B6, B11, B1 i B20 OM. Šta se dešava u KM?

KM	brojači
B10	3
B3	4
B15	1
B1	0
B6	5
B9	10
B12	6
B5	2
B32	7
B4	12

B6

KM	brojači
B10	4
B3	5
B15	2
B1	1
B6	0
B9	10
B12	6
B5	3
B32	7
B4	12

B11

KM	brojači
B10	5
B3	6
B15	3
B1	2
B6	1
B9	11
B12	7
B5	4
B32	8
B11	0

Univerzitet Singidunum www.singidunum.ac.rs

LRU tehnika

Primer 8 - nastavak

Procesor zahteva podatke iz blokova B6, B11, B1 i B20 OM. Šta se dešava u KM?

B11

KM	brojači
B10	5
B3	6
B15	3
B1	2
B6	1
B9	11
B12	7
B5	4
B32	8
B11	0

B1

KM	brojači
B10	5
B3	6
B15	3
B1	0
B6	2
B9	11
B12	7
B5	4
B32	8
B11	1

B20

KM	brojači
B10	6
B3	7
B15	4
B1	1
B6	3
B20	0
B12	8
B5	5
B32	9
B11	2

Tehnike ažuriranja

- Bave se problemom koherencije keša.
- Koherencija podrazumeva usaglašenost između blokova u KM i njihovih kopija u OM.

Vrste tehnika

- ✓ Tehnika upisa ako blok postoji u KM
- ✓ Tehnika upisa ako blok ne postoji u KM
- ✓ Tehnika čitanja ako blok postoji u KM (samo se pročita vrednost)
- ✓ Tehnika čitanja ako blok ne postoji u KM

Tehnika upisa ako blok postoji u KM

Postoje dve mogućnosti upisa:

- trenutni upis (*writte-through*)
  → svakom operacijom upisa se istovremeno vrši upis i u KM i u OM

- odloženi upis (*writte-back*)
  → upis se radi samo u KM, dok je upis u OM odložen do trenutka kada se javi potreba za zamenom bloka
  → svakom bloku u KM pridružen je bit (*dirty bit*) čija vrednost pokazuje da li je postojao bar jedan upis u taj blok od kada je on došao u KM
  → u trenutku zamene, ispituje se bit, pa ako je 1, blok KM se upisuje u OM, a ako je 0, samo se dolazeći blok prepiše preko bloka KM

Tehnika upisa ako blok ne postoji u KM

Postoje dve mogućnosti upisa:

- dovuci blok (*writte-allocate*)
  → blok se dovlači iz OM u KM i onda se radi kao u prethodnoj tehnici

- ne dovlači blok (*writte-no-allocate*)
  → blok se ne dovlači iz OM, već se upis vrši samo u OM

Tehnika čitanja ako blok ne postoji u KM

Postoje dve mogućnosti čitanja:

- direktno prosleđivanje
  → blok se dovlači iz OM u KM, a traženi podatak se odmah po pristizanju u KM prosleđuje procesoru

- prosleđivanje sa zadrškom
  → blok se dovlači iz OM u KM, pa kad se ceo smesti, onda se traženi podatak prosleđuje u procesor