RAČUNALNIŠKA ARHITEKTURA

9 Pomnilniška hierarhija

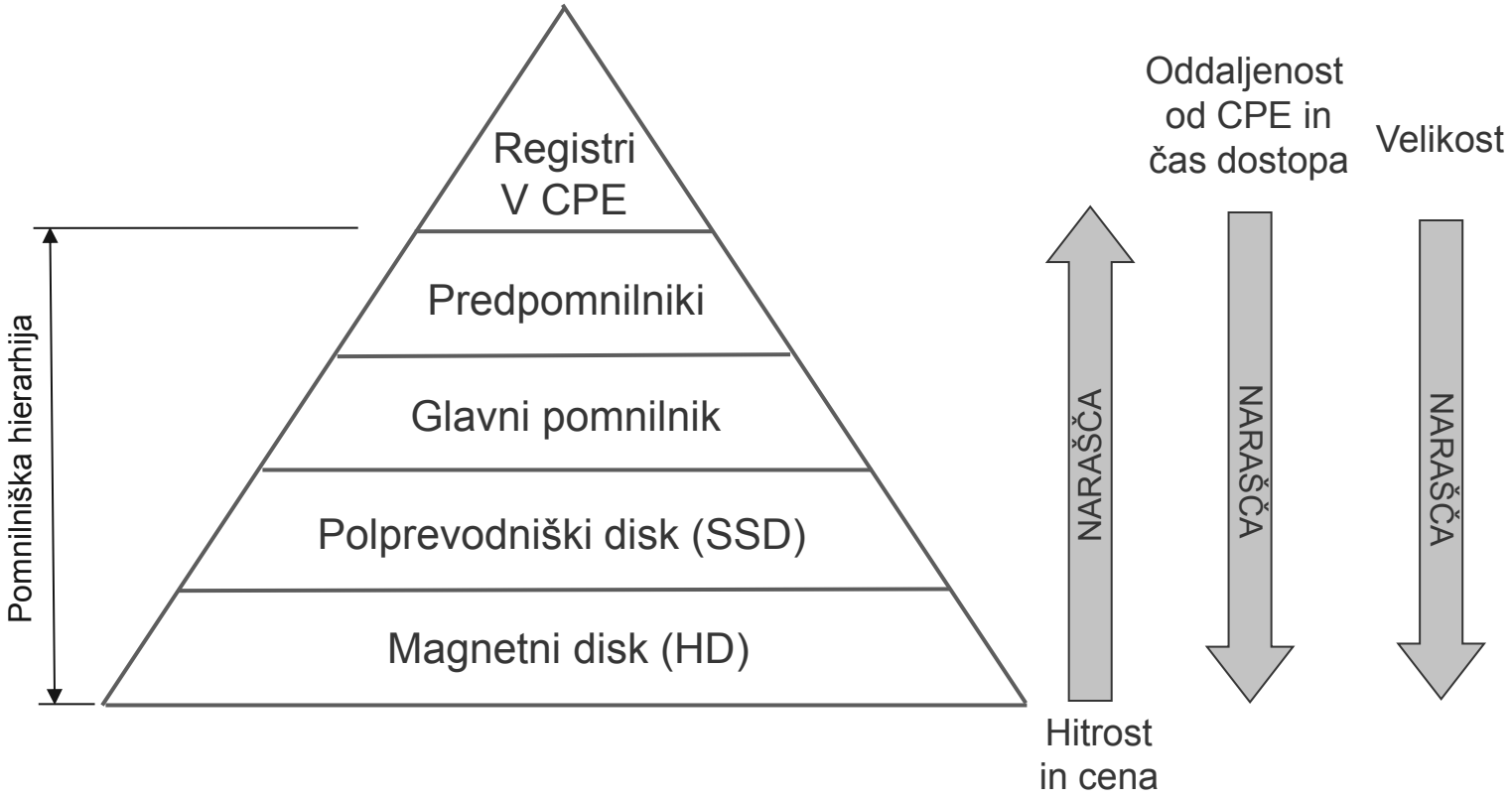
9 Pomnilniška hierarhija

- Predpomnilnik
  - Primer delovanja predpomnilnika
  - Vrste predpomnilnikov glede na omejitve pri preslikavi blokov
  - Vpliv predpomnilnika na hitrost delovanja CPE
  - Primer: Vpliv predpomnilnika L2 na hitrost CPE
- Navidezni pomnilnik
  - Navidezni pomnilnik z ostranjevanjem
  - Napake strani
  - Pohitritev preslikovanja
  - Strategije in algoritmi
- Delovanje pomnilniške hierarhije
  - Štirinivojska pomnilniška hierarhija – povprečni čas dostopa kot ga vidi CPE
  - Vpliv verjetnosti zgrešitve v glavnem pomnilniku na povprečni dostopni čas pri trinivojski pomnilniški hierarhiji

Pomnilniška hierarhija

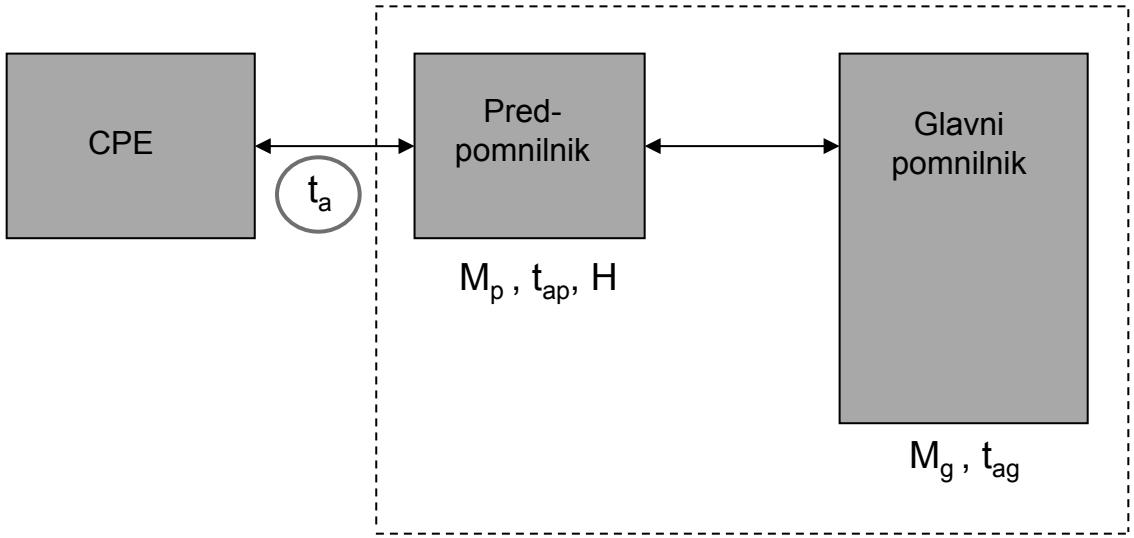
- Želja programerjev: neomejena količina hitrega pomnilnika.
- Pomnilniška hierarhija, ki jo sestavlja več ločenih pomnilnikov z različnimi lastnostmi, omogoča realizacijo te iluzije.
  - Predpomnilnik (lahko več nivojev predpomnilnikov)
  - Glavni pomnilnik
  - Navidezni pomnilnik
- Uspešno delovanje pomnilniške hierarhije je možno zaradi lokalnosti pomnilniških dostopov (glej RA-3/101).
  - Časovna lokalnost
  - Krajevna lokalnost

Pomnilniki, ki sestavljajo pomnilniško hierarhijo



9.1 Predpomnilnik

- Predpomnilnik (cache) je majhen hiter pomnilnik (SRAM) med CPE in glavnim pomnilnikom.
- Uporaba predpomnilnika v pomnilniški hierarhiji ustvari iluzijo hitrega pomnilnika.
- Vsebina predpomnilnika: podmnožica vsebine glavnega pomnilnika.
- CPE s pomnilniškim naslovom vedno dostopa do predpomnilnika.



Dvonivojska pomnilniška hierarhija

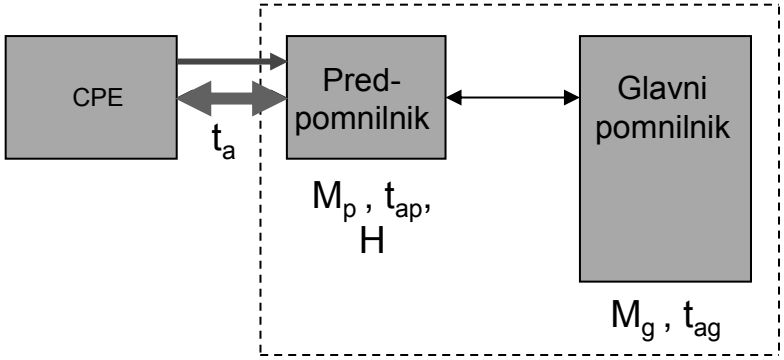
M_p predpomnilnik
t_{ap} čas dostopa do predpomnilnika [ns]
H verjetnost zadetka v predpomnilniku [%]

M_g glavni pomnilnik
t_{ag} čas dostopa do glavnega pomnilnika [ns]

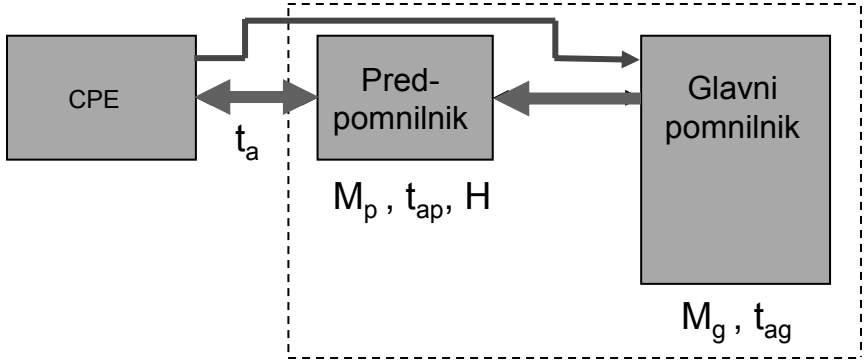
t_a povprečni čas dostopa do celotne hierarhije, kot ga vidi CPE [ns]

- Pri dostopu CPE do informacije (ukaz, operand) v predpomnilniku sta dve možnosti:
  - **Zadetek** (hit), če je naslov (in vsebina s tega naslova) v predpomnilniku ⇒ čas dostopa je t_{ap}
  - **Zgrešitev** (miss), če naslova (in vsebine) ni v predpomnilniku ⇒ čas dostopa je $t_{ap} + t_{ag}$

Zadetek v predpomnilniku



Zgrešitev v predpomnilniku



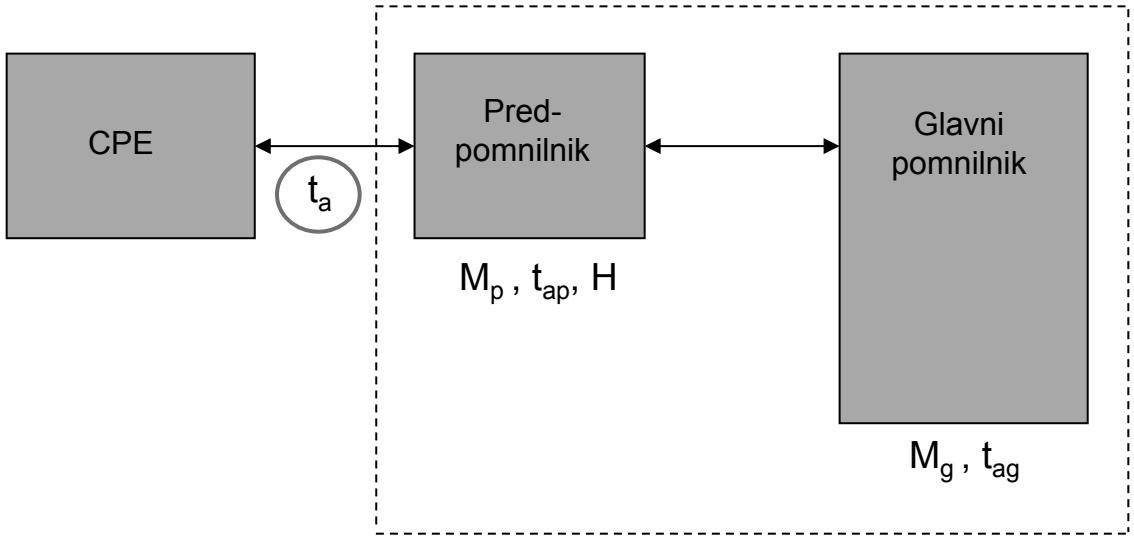
- Uspešnost delovanja predpomnilnika merimo:
  - Z verjetnostjo zadetka *H* $$H = \frac{N_p}{N} = \frac{N_p}{N_g + N_p}$$

    N - število vseh dostopov do predpomnilnika ($N = N_g + N_p$)

    N_p - število zadetkov (želena informacija je v predpomnilniku)

    N_g - število zgrešitev (želene informacije ni v predpomnilniku, potreben je prenos informacije iz glavnega pomnilnika v predpomnilnik)

  - Ali z verjetnostjo zgrešitve *1 - H* (želimo čim manjšo)
- V predpomnilniku je verjetnost zadetka H > 0,9 (90%)
- V primeru zgrešitve je potreben dostop do glavnega pomnilnika.



- Povprečni dostopni čas t_a, kot ga vidi CPE, je:

$$t_a = t_{ap} + (1 - H)t_{ag}$$

- Pri računanju je treba upoštevati dve posebnosti predpomnilnikov:
  - Med glavnim pomnilnikom in predpomnilnikom se vedno prenaša **predpomnilniški blok**, kar je več sosednjih besed
  - Čas v računalniku merimo z urinimi periodami
- Čas t_{ap} je na nivoju L1 pri večini računalnikov od ene do nekaj urinih period.
- Pri zgrešitvi $\Rightarrow$ dostop do glavnega pomnilnika in prenos bloka v predpomnilnik $\Rightarrow$ imenujemo skupni čas **zgrešitvena kazen t_B.**

- Zgrešitvena kazen t_B je čas, ki se ob zgrešitvi (verjetnost za zgrešitev je $1 - H$) prišteje k času dostopa do predpomnilnika..
- Zgrešitvena kazen je tipično med 10 in 100 urinimi periodami.
- Če ima računalnik tudi predpomnilnik na nivoju L2, je zgrešitvena kazen precej manjša, ker je predpomnilnik L2 hitrejši kot glavni pomnilnik.

- Povprečni čas dostopa t_a je z upoštevanjem zgrešitvene kazni:

$$t_a = t_{ap} + (1 - H)t_B$$

t_{ap} – dostopni čas do predpomnilnika
(1 - H) - verjetnost zgrešitve v predpomnilniku
t_B – zgrešitvena kazen (čas dostopa do glavnega pomnilnika + čas za prenos bloka v predpomnilnik)

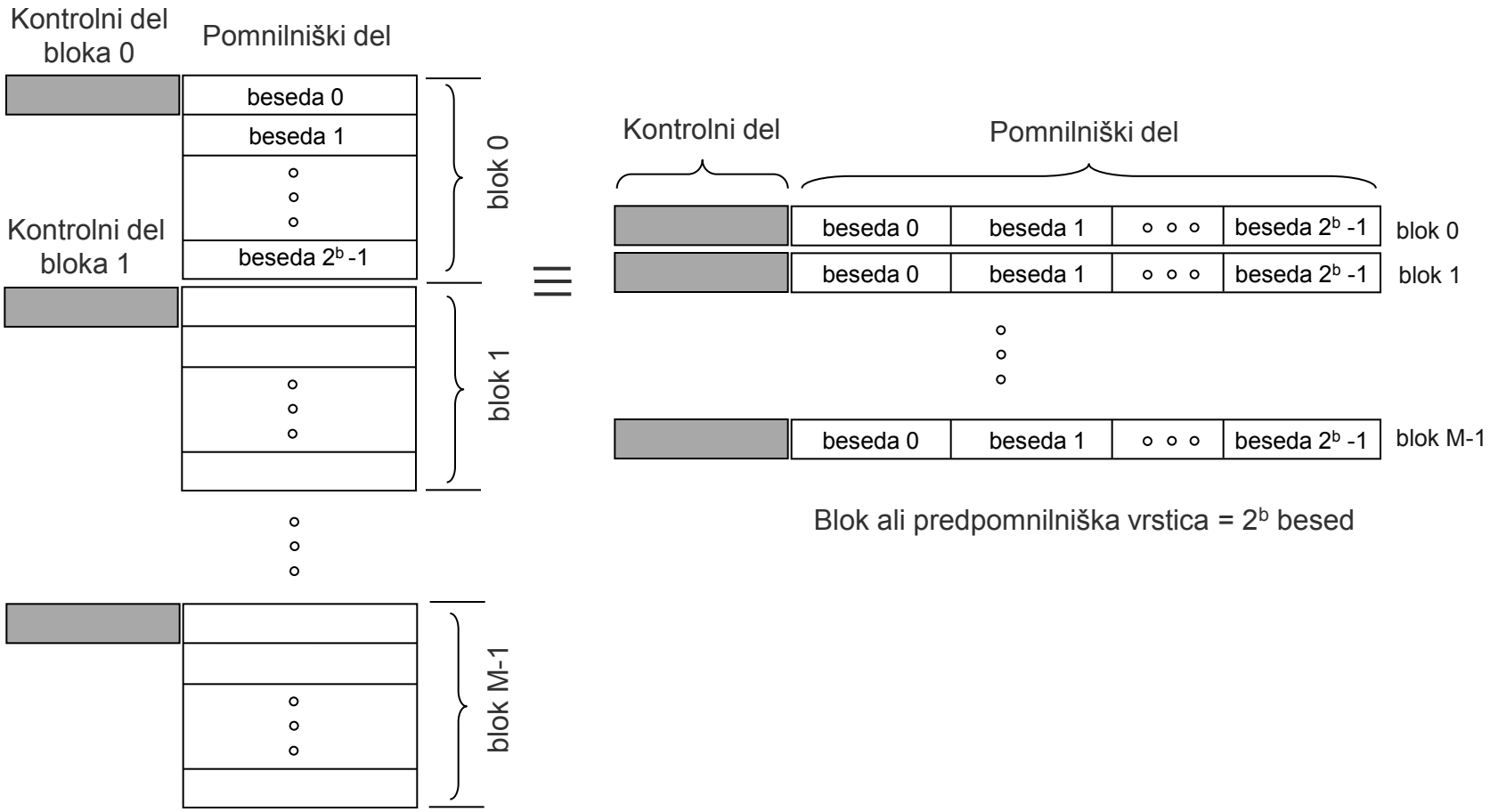
- Če sta časa t_{ap} in t_B v urinih periodah, je seveda tudi rezultat t_a v urinih periodah.

- Povprečni dostopni čas v sekundah (t_{CPE} je trajanje ene urine periode v sekundah) :

t_a [s] = t_a [urine periode] * t_{CPE} [s]

- Vsebina v predpomnilniku se spreminja $\Rightarrow$
  - Bloki iz glavnega pomnilnika
  - Naslovi blokov (številke blokov iz glavnega pomnilnika)
- Zato je vsak predpomnilnik sestavljen iz dveh delov:
  - **Pomnilniškega dela**, ki je razdeljen na bloke ali predpomnilniške vrstice
  - **Kontrolnega dela**, ki ga sestavljajo kontrolne besede. Vsakemu bloku v pomnilniškem delu pripada ena kontrolna beseda, ki vsebuje naslov bloka (številko bloka v glavnem pomnilniku)

Zgradba predpomnilnika

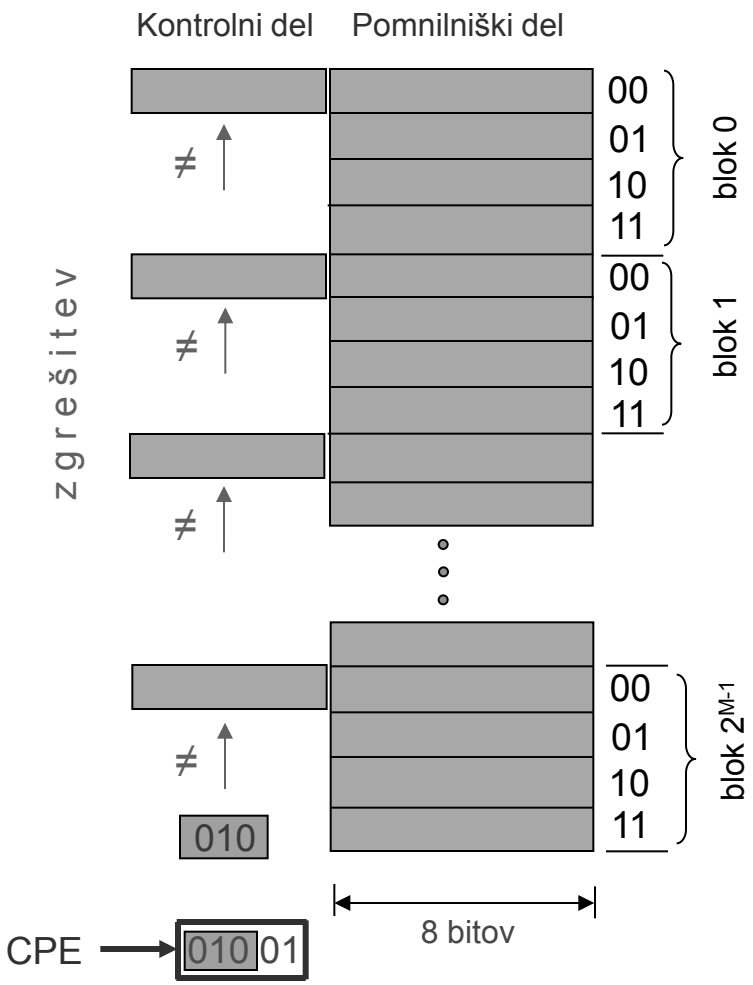


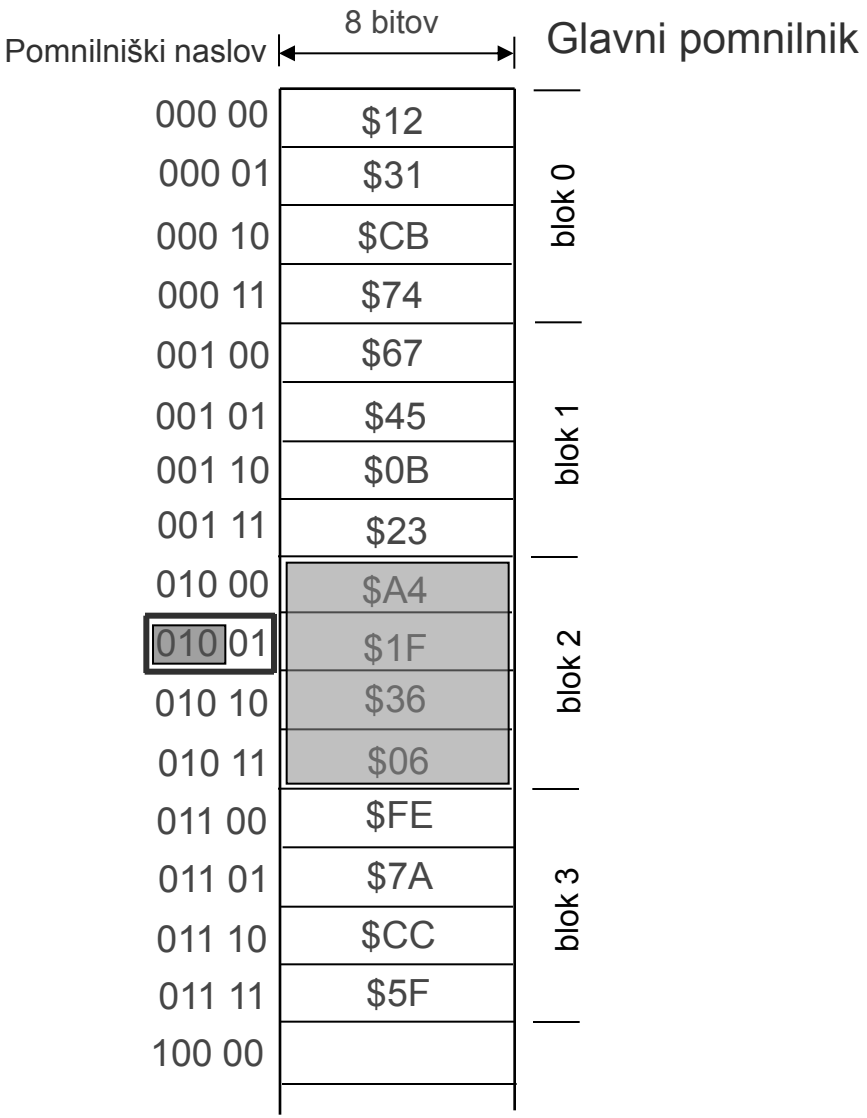
- Blok (ali predpomnilniška vrstica) je nekaj sosednjih pomnilniških besed.
- Velikost bloka ($B = 2^b$) je tipično 4 do 512 pomnilniških besed.
- Med glavnim pomnilnikom in predpomnilnikom se **vedno prenaša cel blok.**
- Ko se blok iz glavnega pomnilnika prenese v prosti blok v predpomnilniku se:
  - Vsebina bloka se prenese v pomnilniški del bloka v predpomnilniku
  - Naslov (številka) bloka se prenese v kontrolni del bloka v predpomnilniku

- Primer delovanja predpomnilnika:
  - Predpostavimo:
    - da procesor dostopa zaporedoma do pomnilniških besed z naslednjimi desetiškimi pomnilniškimi naslovi:
    - 9, 10, 11, 2, 3, 9, 10, 11, 2, … ;
    - da ima predpomnilnik bloke velikosti 4 bajte ($B = 2^2 = 4$) in je na začetku prazen;
    - da je pomnilniški naslov je dolg 5 bitov.
    - Potem zgornji trije biti pomnilniškega naslova določajo številko bloka, spodnja dva bita pomnilniškega naslova pa besedo v bloku ($2^2 = 4$)

CPE dostopa do pomn. naslova:

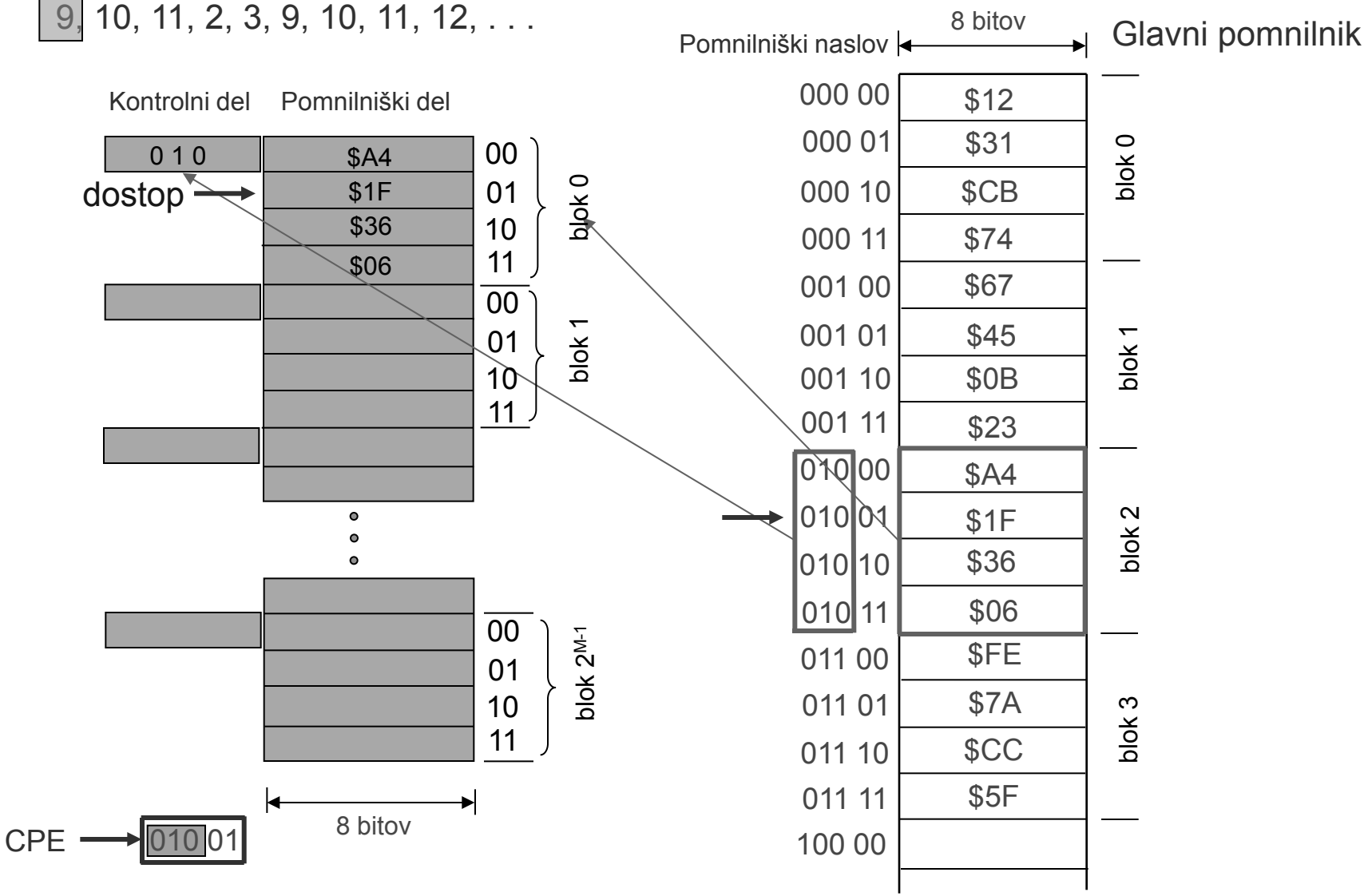
9, 10, 11, 2, 3, 9, 10, 11, 12, . . .



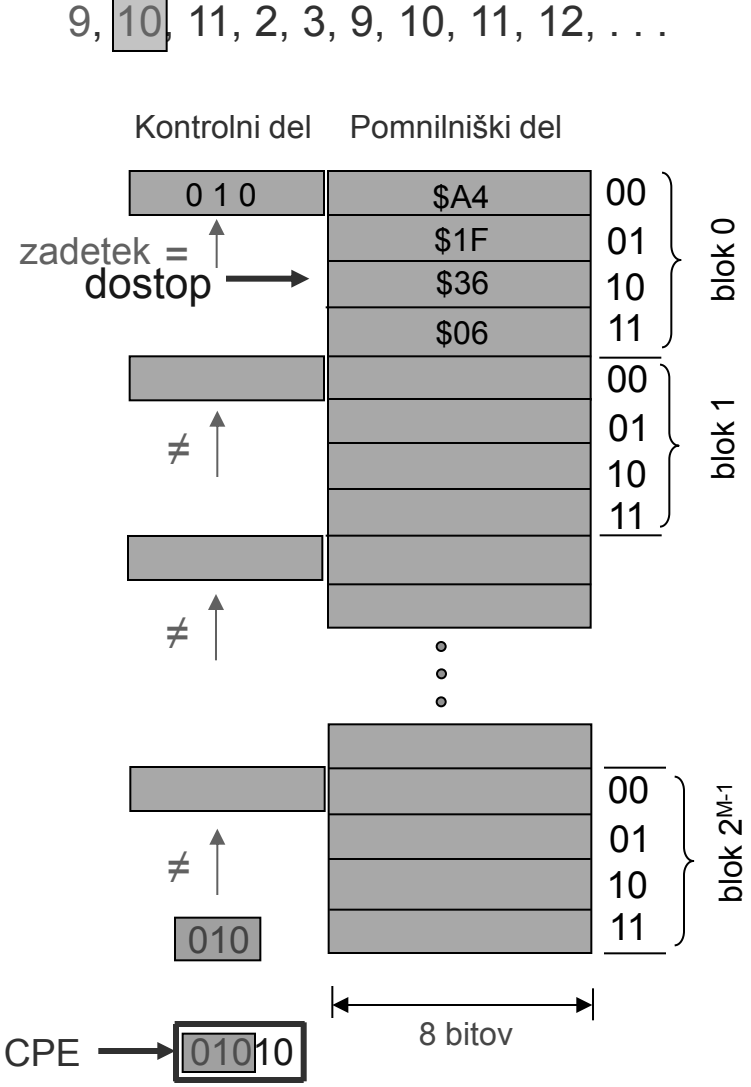


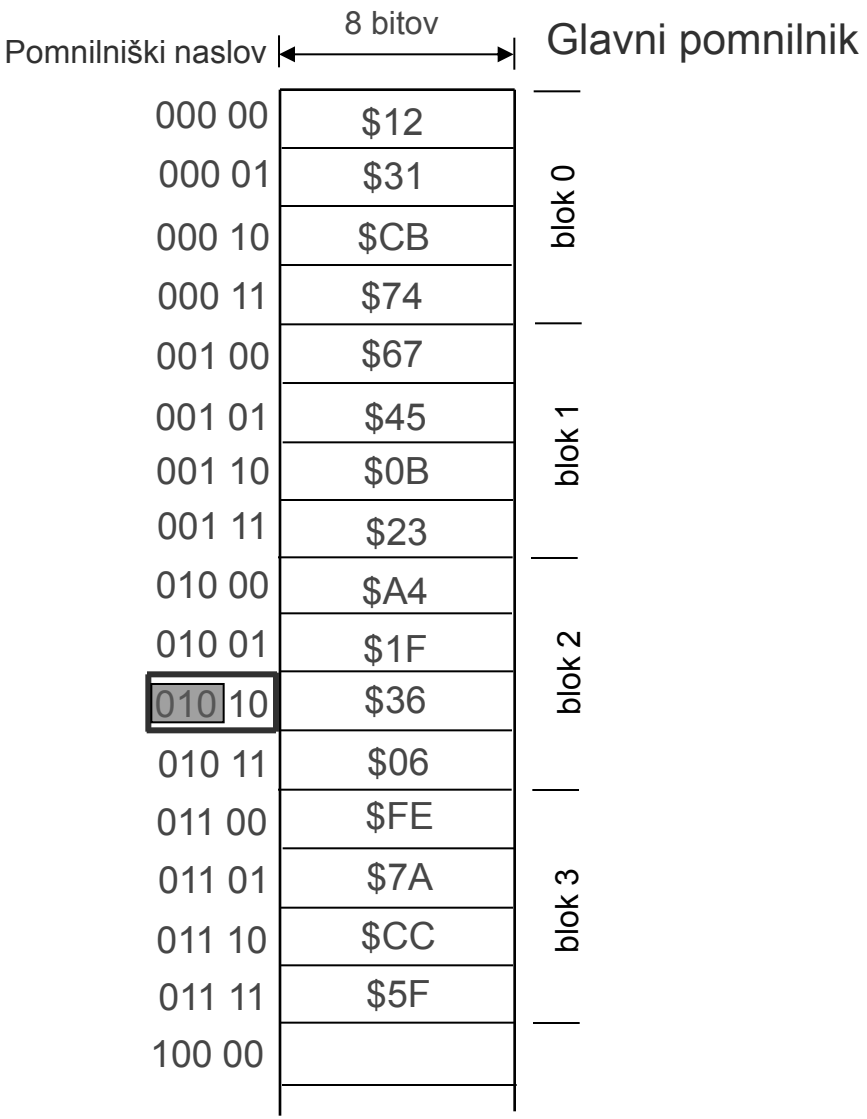
CPE dostopa do pomn. naslova:

9, 10, 11, 2, 3, 9, 10, 11, 12, . . .

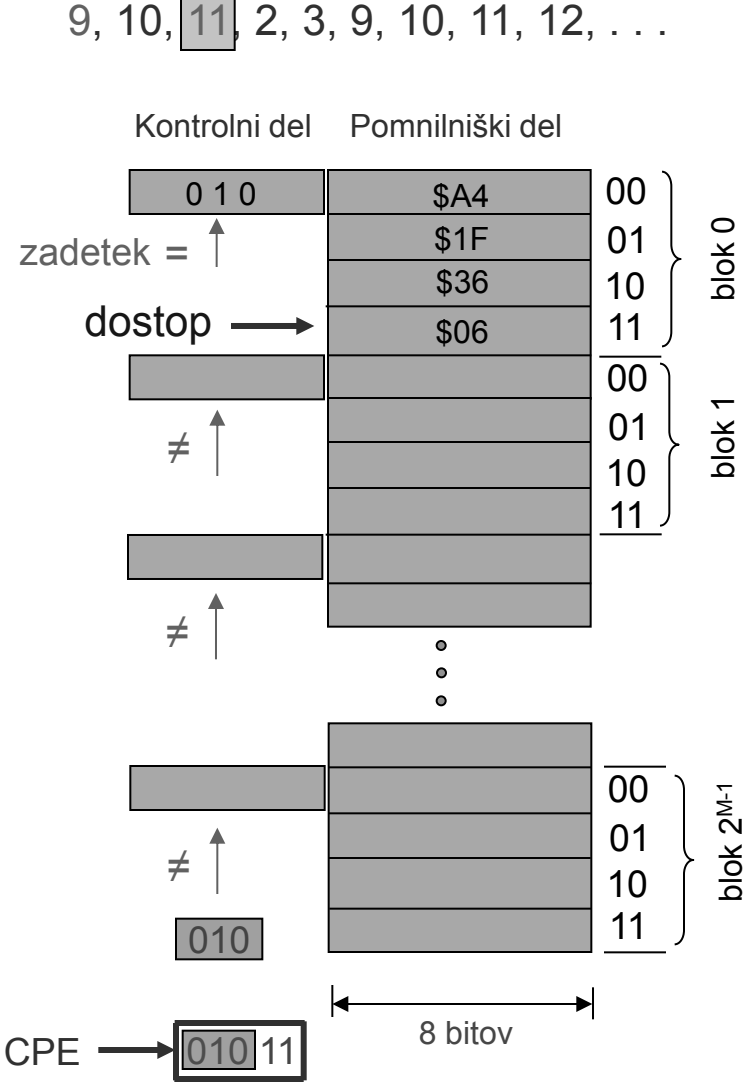


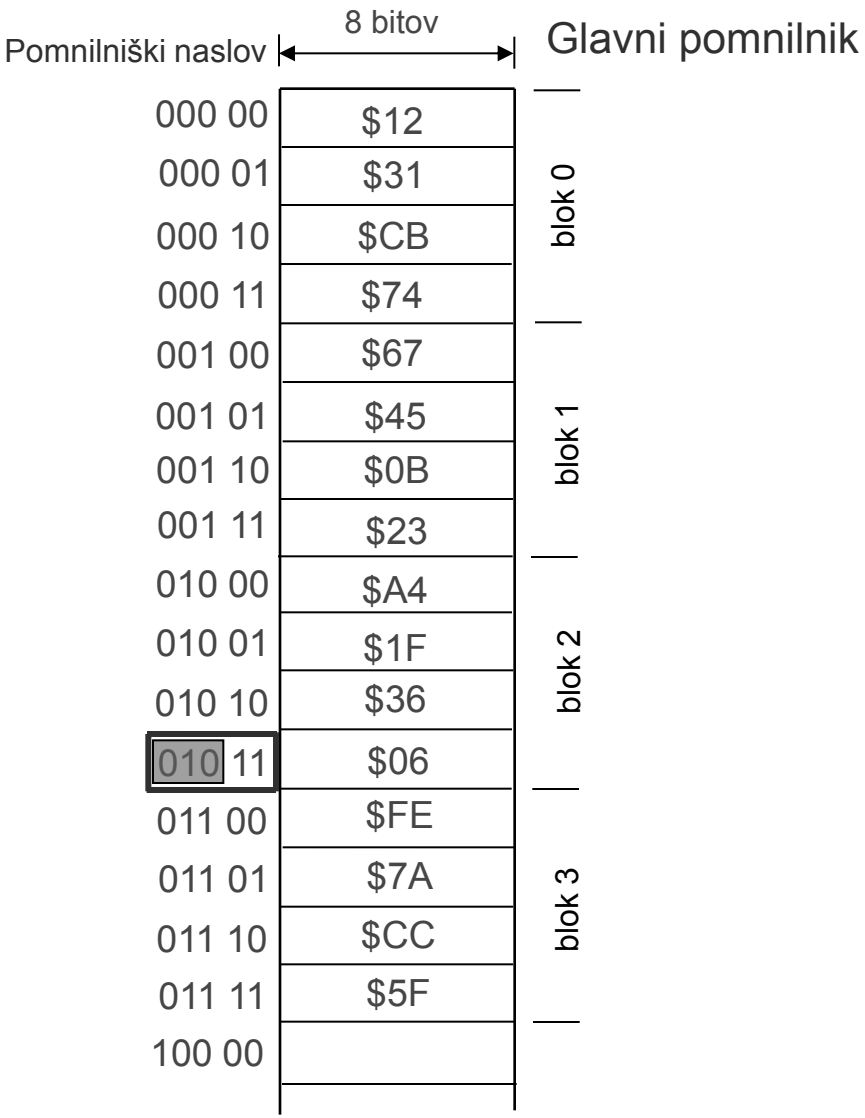
CPE dostopa do pomn. naslova:
9, 10, 11, 2, 3, 9, 10, 11, 12, . . .



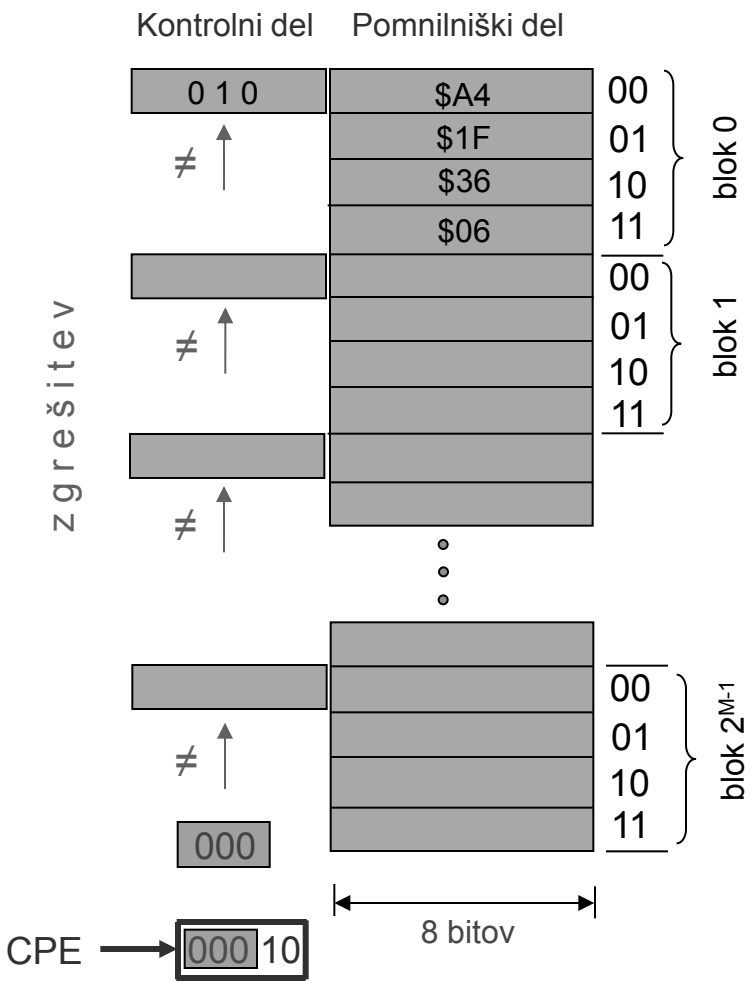


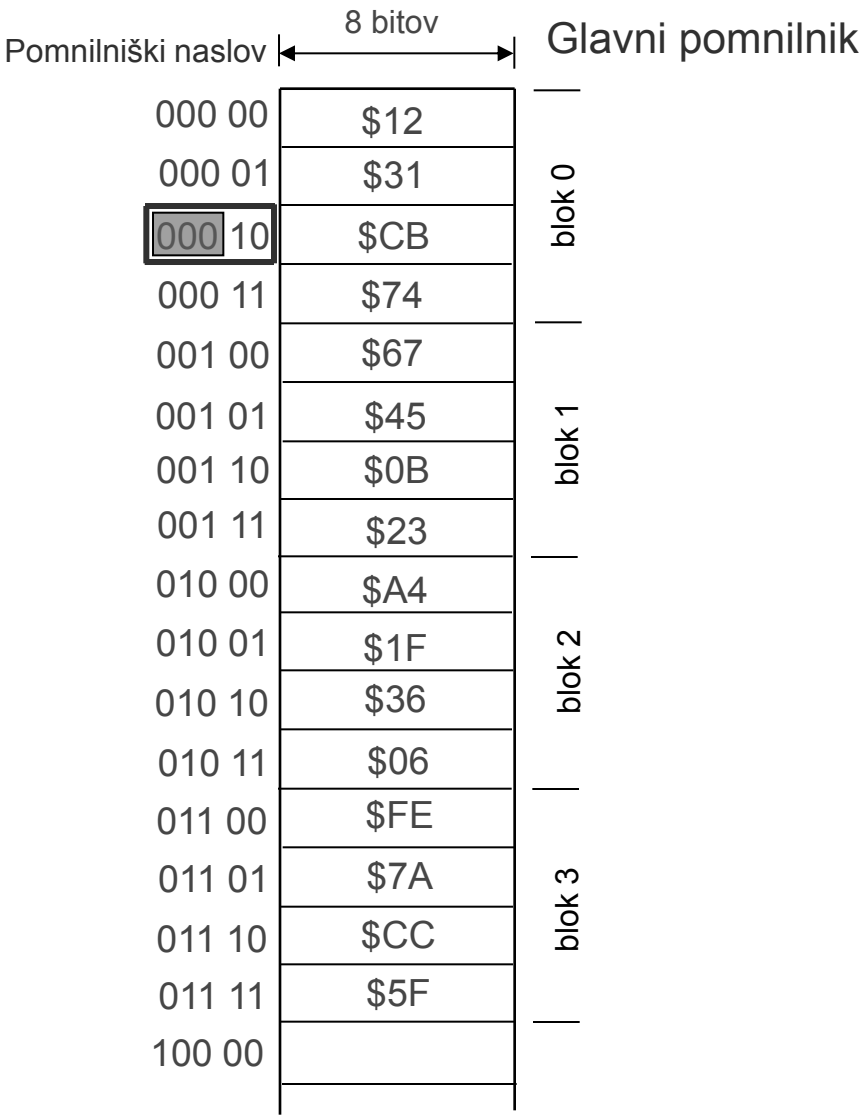
CPE dostopa do pomn. naslova:
9, 10, 11, 2, 3, 9, 10, 11, 12, . . .





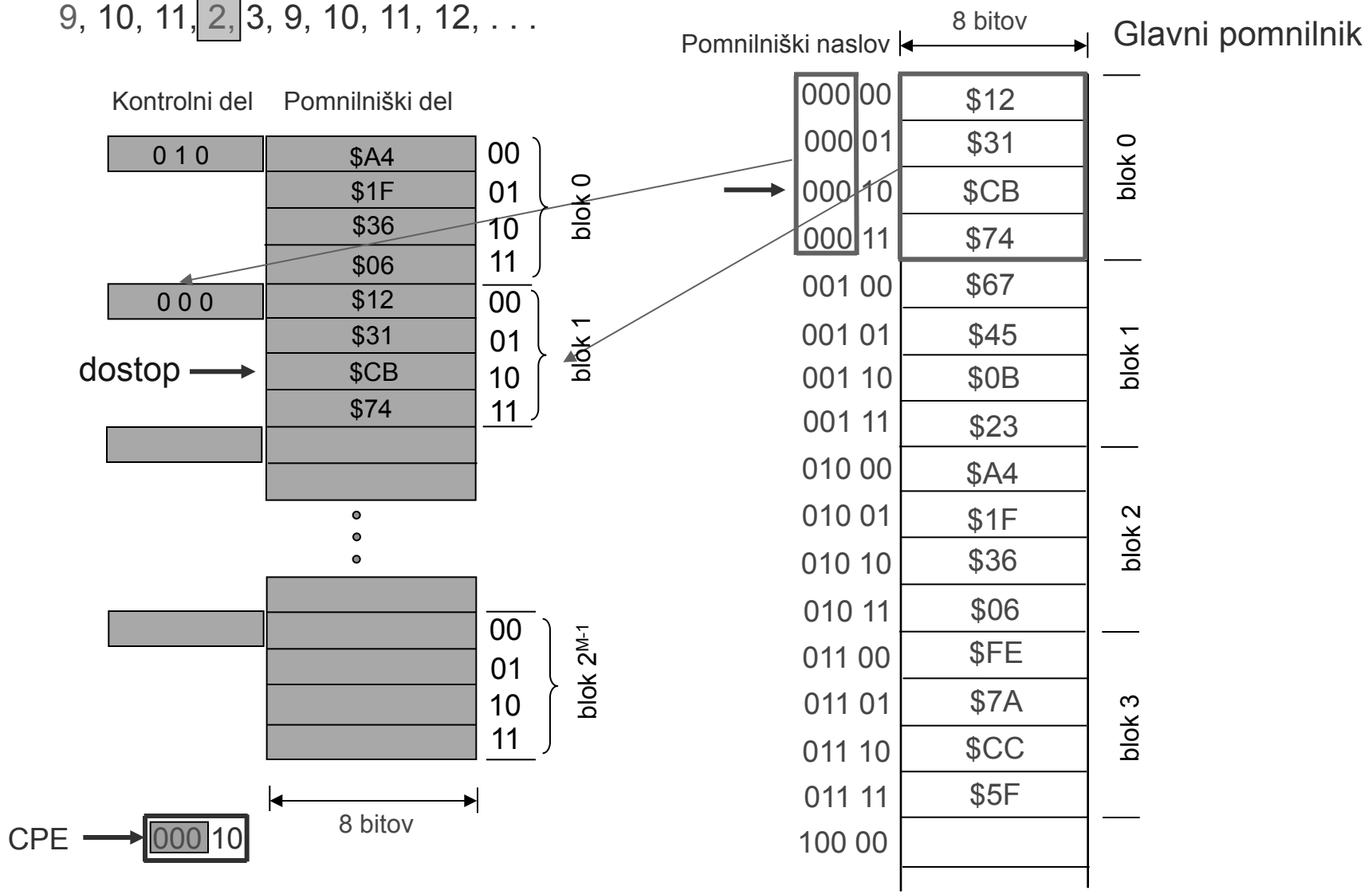
CPE dostopa do pomn. naslova:
9, 10, 11, 2, 3, 9, 10, 11, 12, . . .





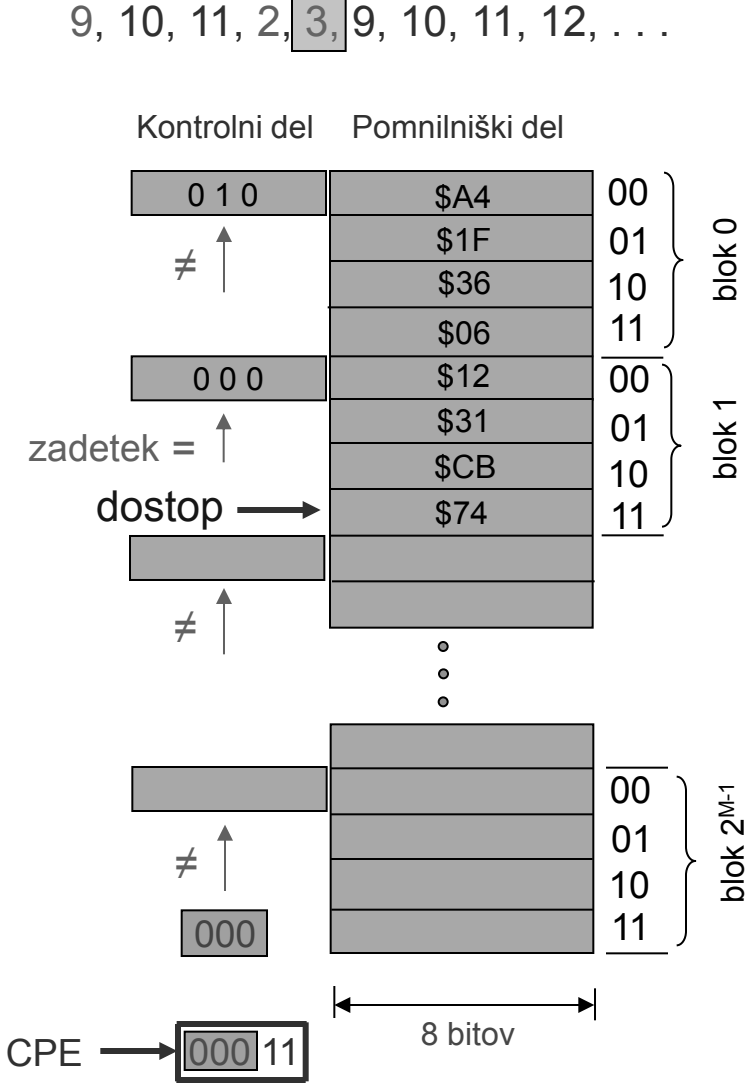
CPE dostopa do pomn. naslova:

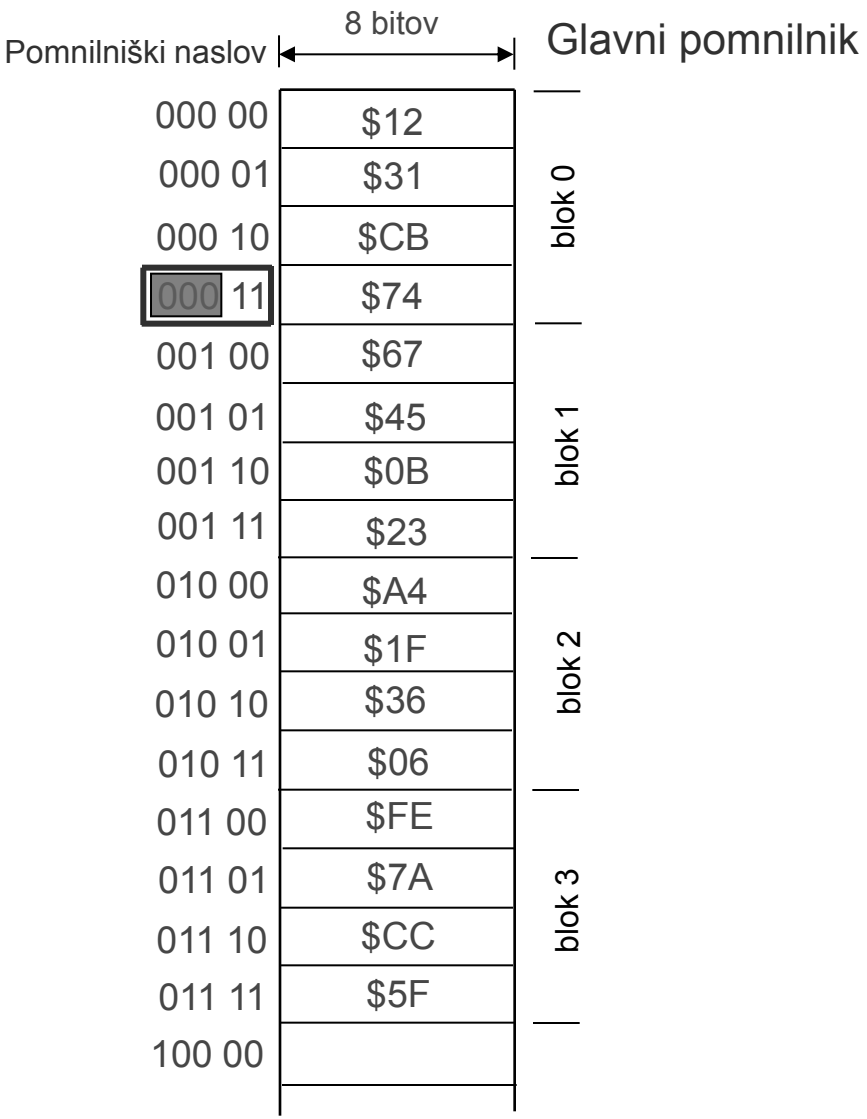
9, 10, 11, 2, 3, 9, 10, 11, 12, . . .



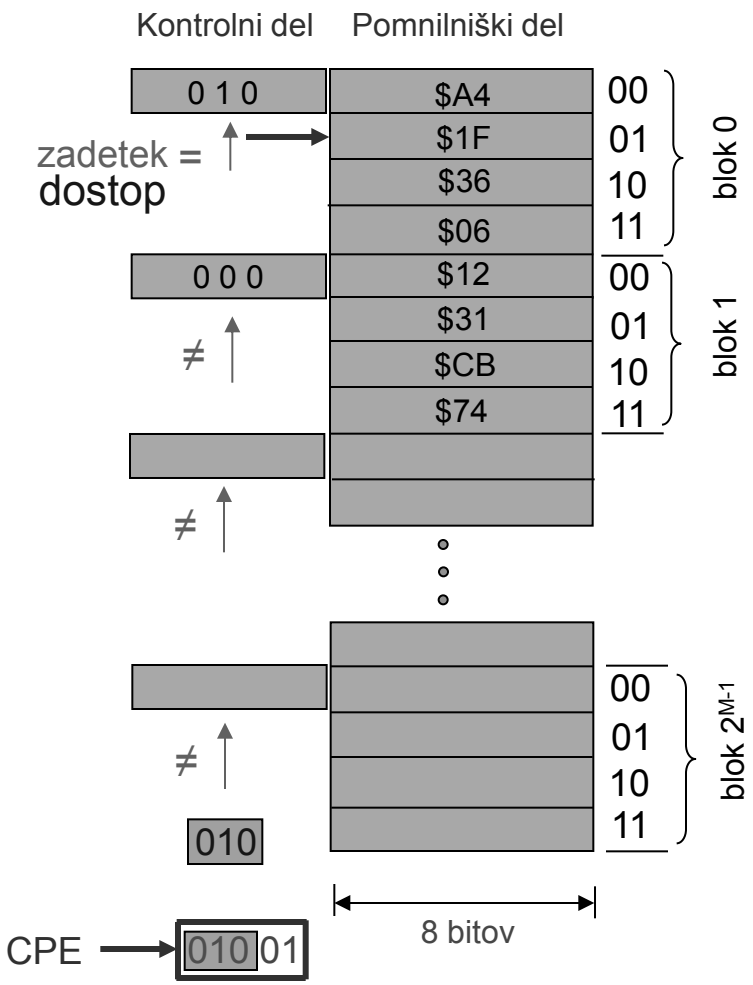
CPE dostopa do pomn. naslova:

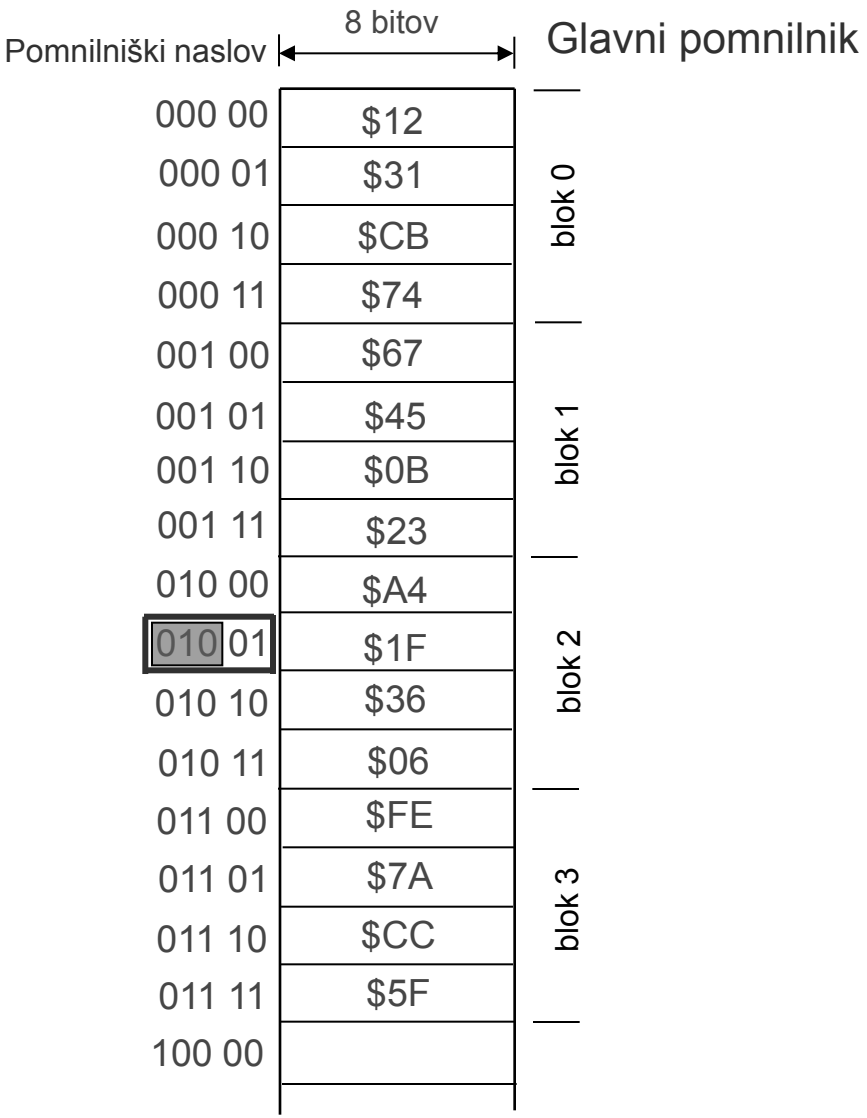
9, 10, 11, 2, 3, 9, 10, 11, 12, . . .





CPE dostopa do pomn. naslova:
9, 10, 11, 2, 3, 9, 10, 11, 12, . . .

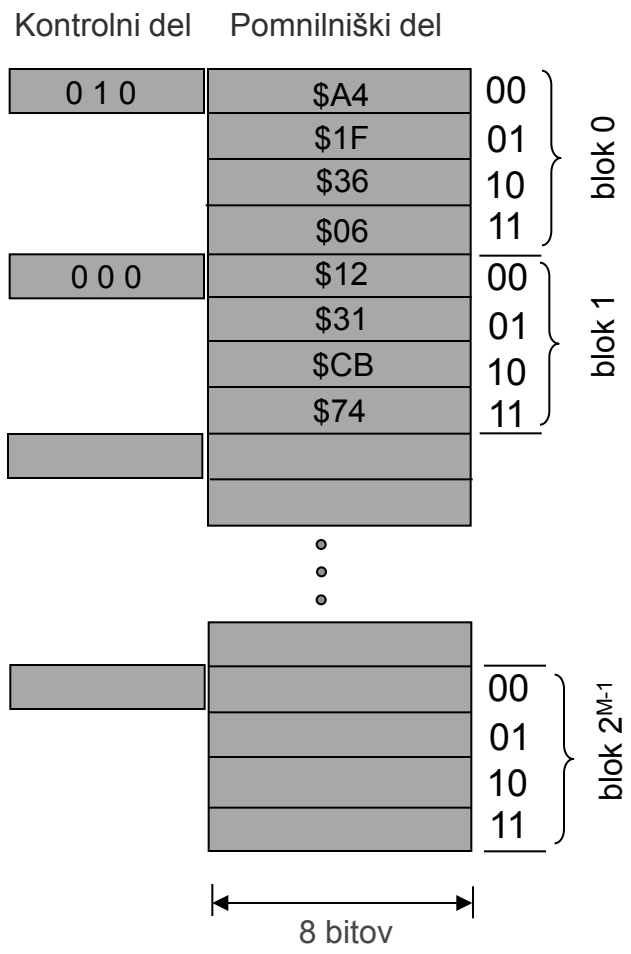


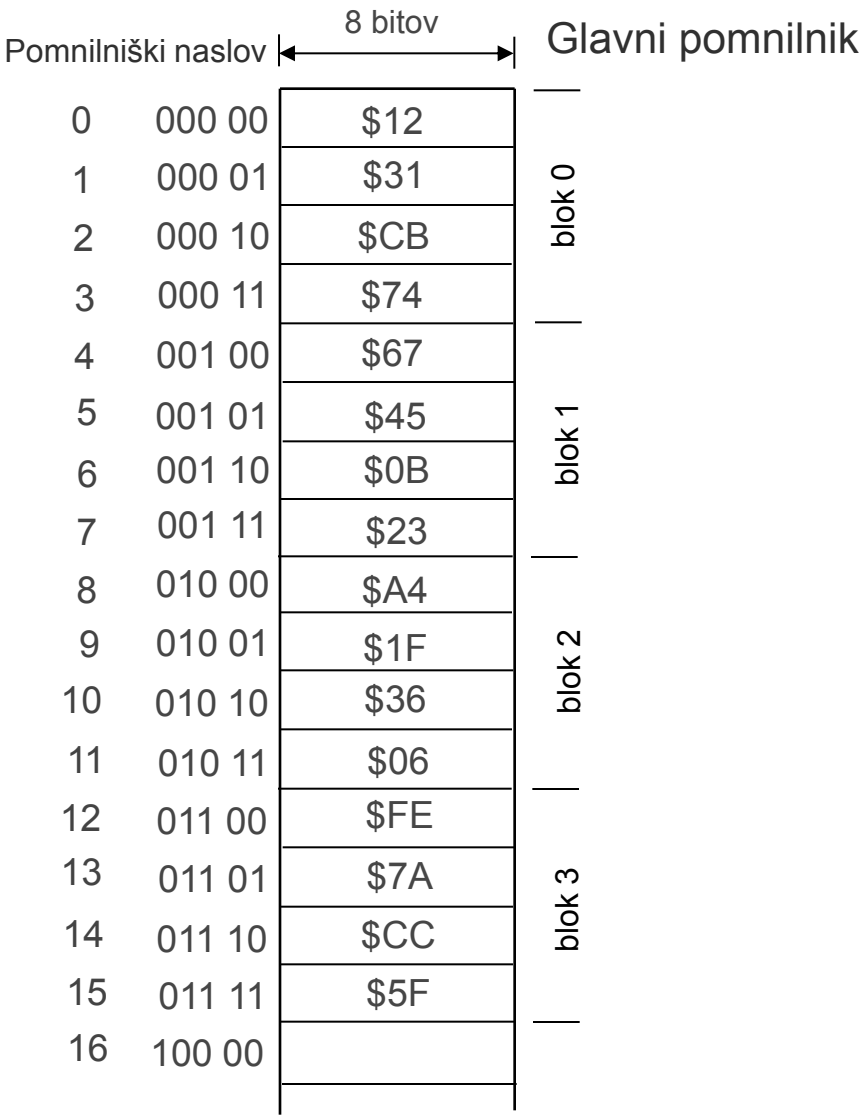


CPE dostopa do pomn. naslova:

9, 10, 11, 2, 3, 9, 10, 11, 2, . . .

zgrešitvi



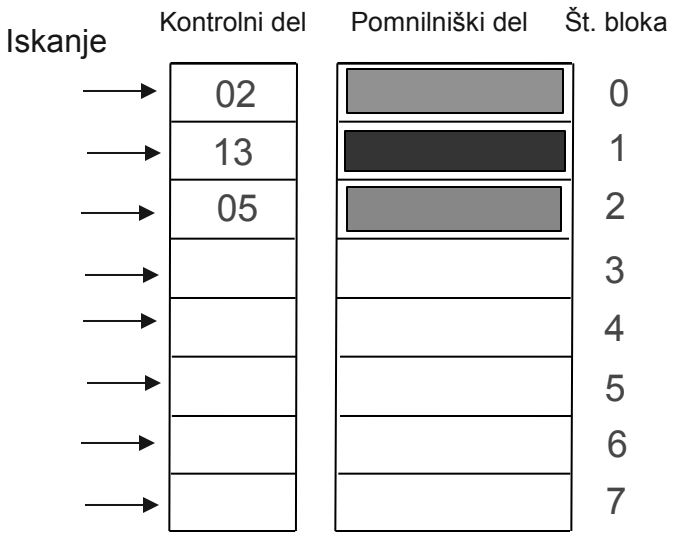


- Zadetek v predpomnilniku (verjetnost H):
  - CPE dostopa do informacije v predpomnilniku (bere ali piše)
- Zgrešitev v predpomnilniku (verjetnost 1 – H):
  - Preslikava bloka iz glavnega pomnilnika v predpomnilnik ali
  - zamenjava bloka – če je predpomnilnik poln, se eden od blokov v predpomnilniku shrani nazaj v glavni pomnilnik (ali je to vedno potrebno?), na njegovo mesto pa se preslika nov blok iz glavnega pomnilnika
  - CPE dostopa do informacije v predpomnilniku

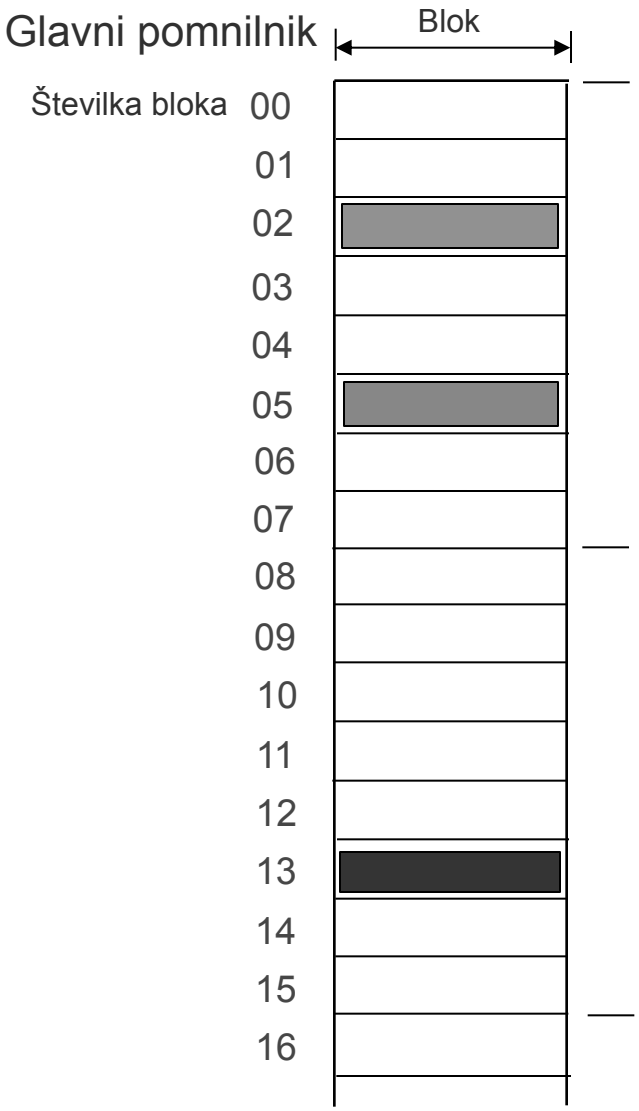
Vrste predpomnilnikov glede na omejitve pri preslikavi blokov

- Iskanje bloka v predpomnilniku mora biti hitro.
- Če to ni mogoče, je treba pri preslikavi bloka iz glavnega pomnilnika v predpomnilnik vpeljati omejitve.
- Glede na strogost omejitev pri preslikavi bloka iz glavnega pomnilnika v predpomnilnik, razlikujemo tri vrste predpomnilnikov:
  - Čisti asociativni predpomnilnik – asociativni kontrolni del
  - Set-asociativni predpomnilnik – asociativen kontrolni del seta
    Število blokov v setu je stopnja asociativnosti E
  - Direktni predpomnilnik – kontrolni del je navadni z naslovom
    naslovljiv pomnilnik

Čisti asociativni predpomnilnik velikosti 8 blokov

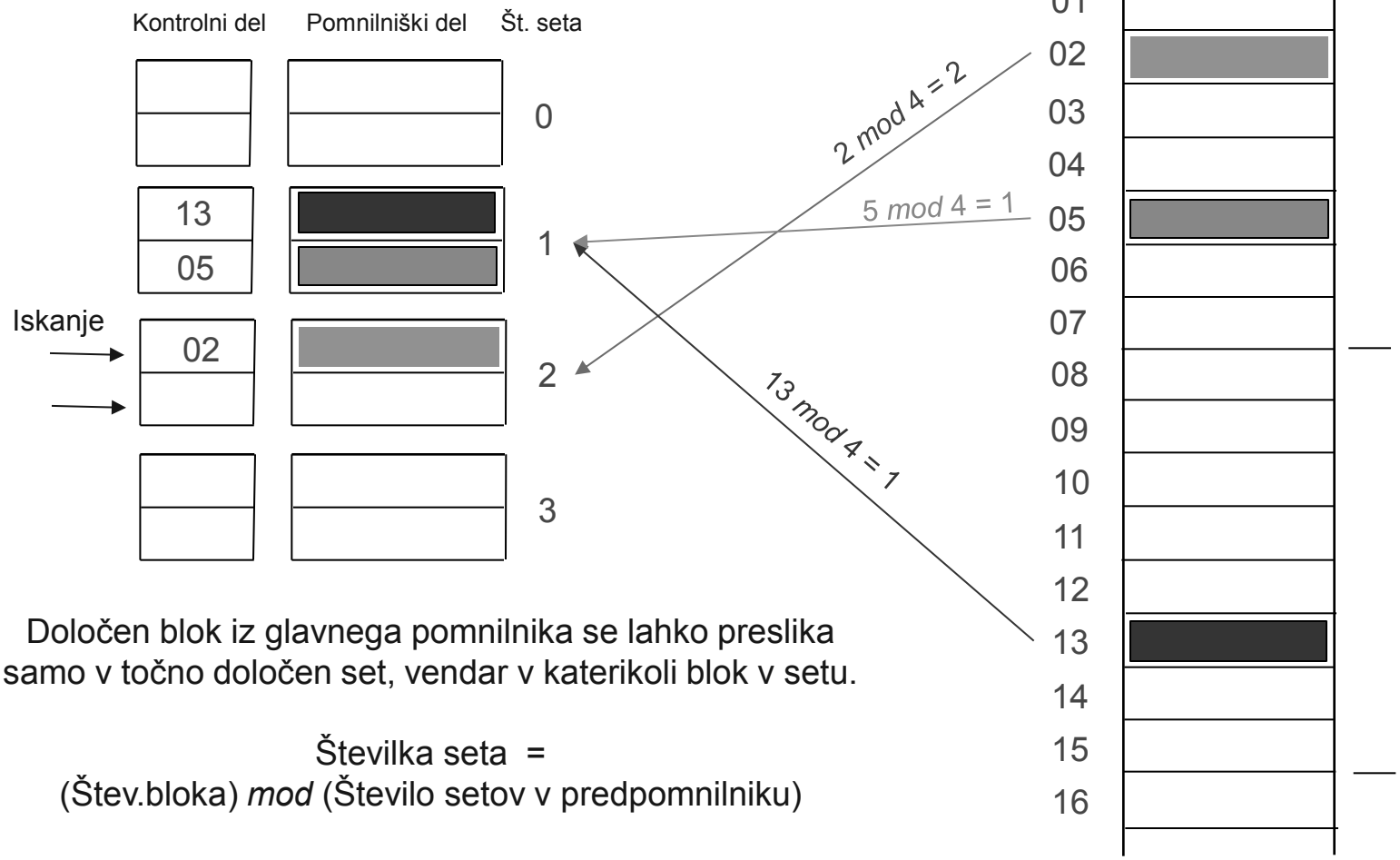


Blok iz glavnega pomnilnika se lahko preslika v katerikoli blok predpomnilnika brez omejitev, ker je iskanje po asociativnem kontrolnem delu hitro.



Preslikava bloka pri set-asociativnem predpomnilniku

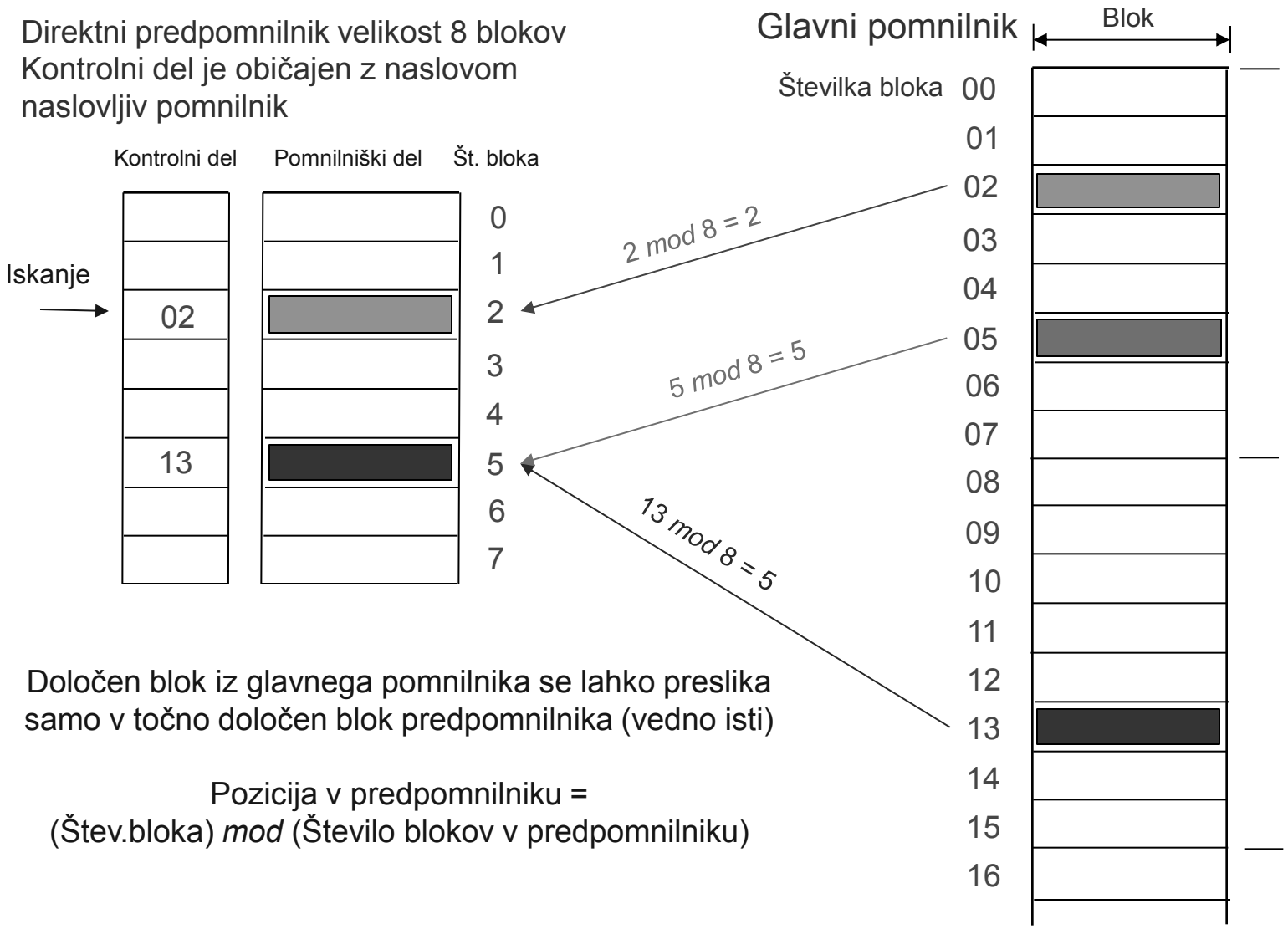
Set-asociativni predpomnilnik
velikost 8 blokov, razdeljen na 4 sete.
2 bloka v setu (= stopnja asociativnosti E=2)



Določen blok iz glavnega pomnilnika se lahko preslika samo v točno določen set, vendar v katerikoli blok v setu.

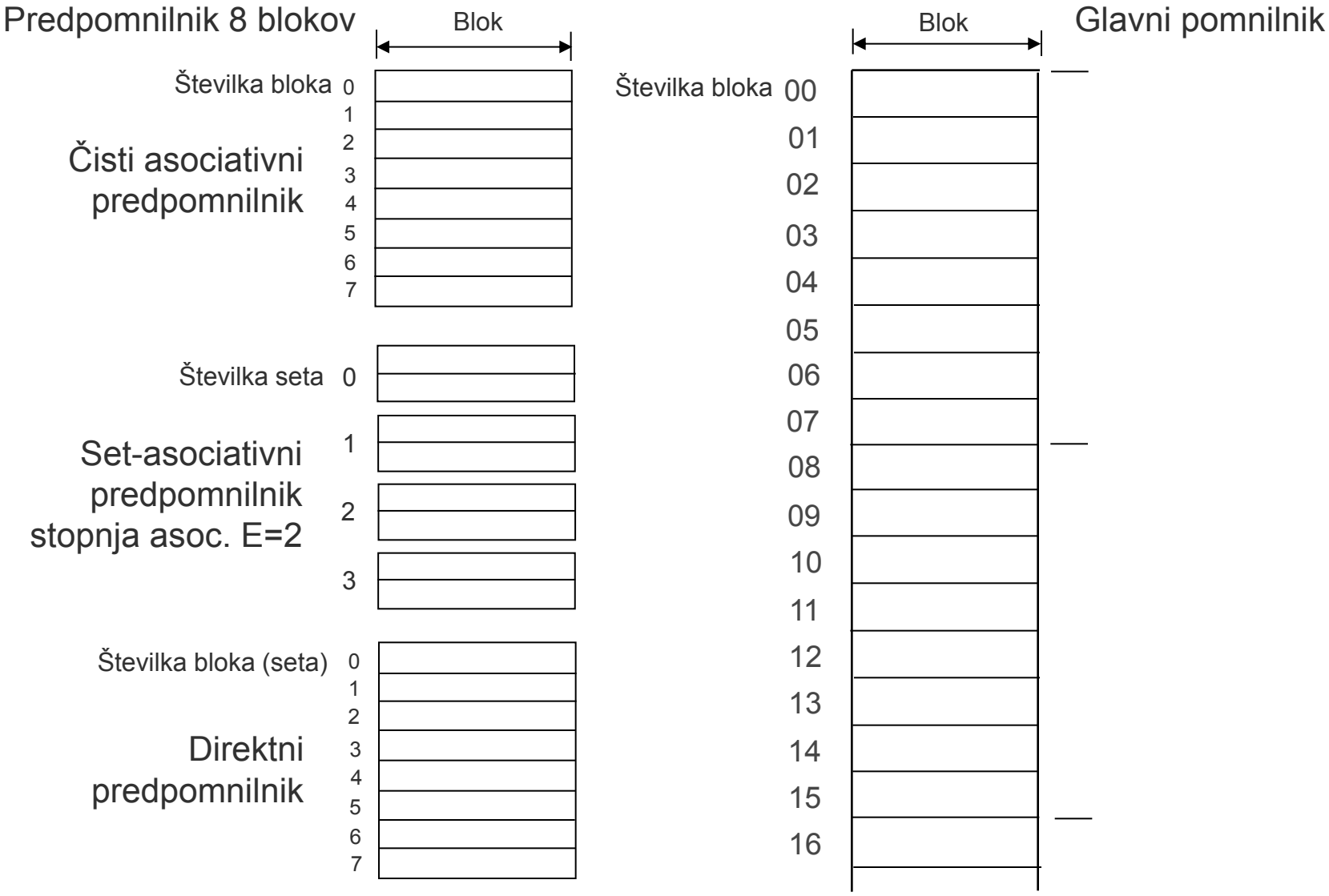
Številka seta =
(Štev.bloka) *mod* (Število setov v predpomnilniku)

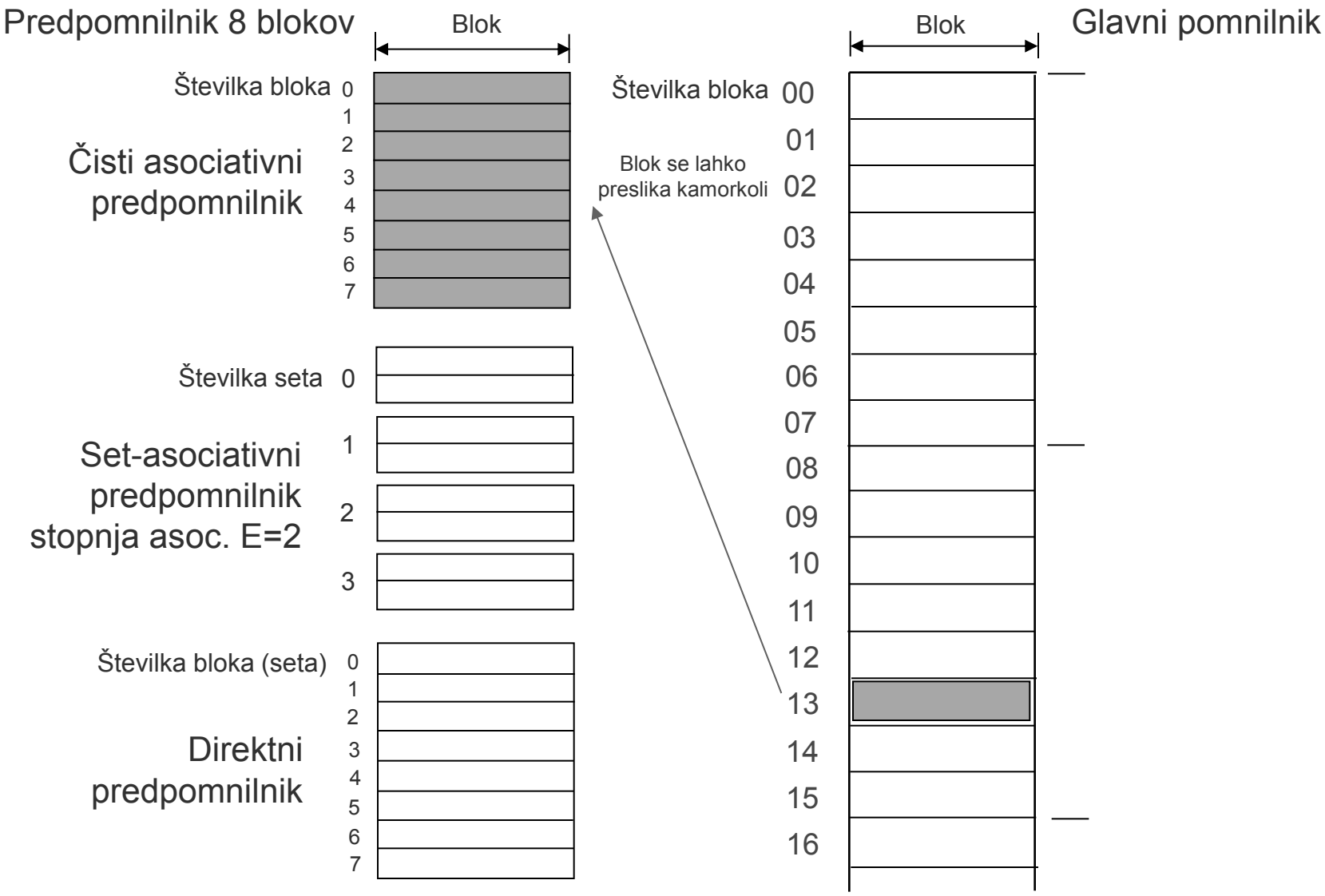
Direktni predpomnilnik velikost 8 blokov
Kontrolni del je običajen z naslovom naslovljiv pomnilnik

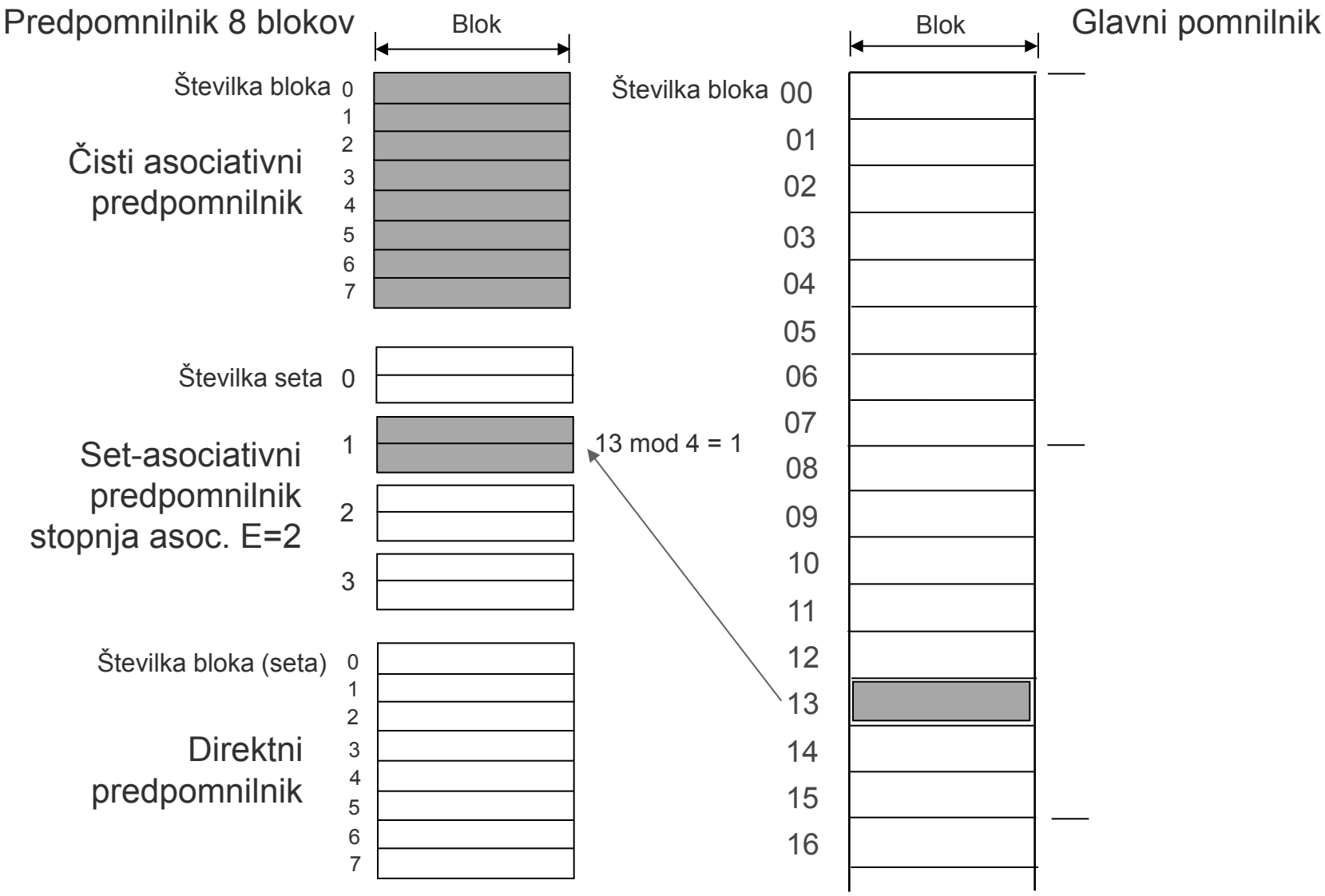


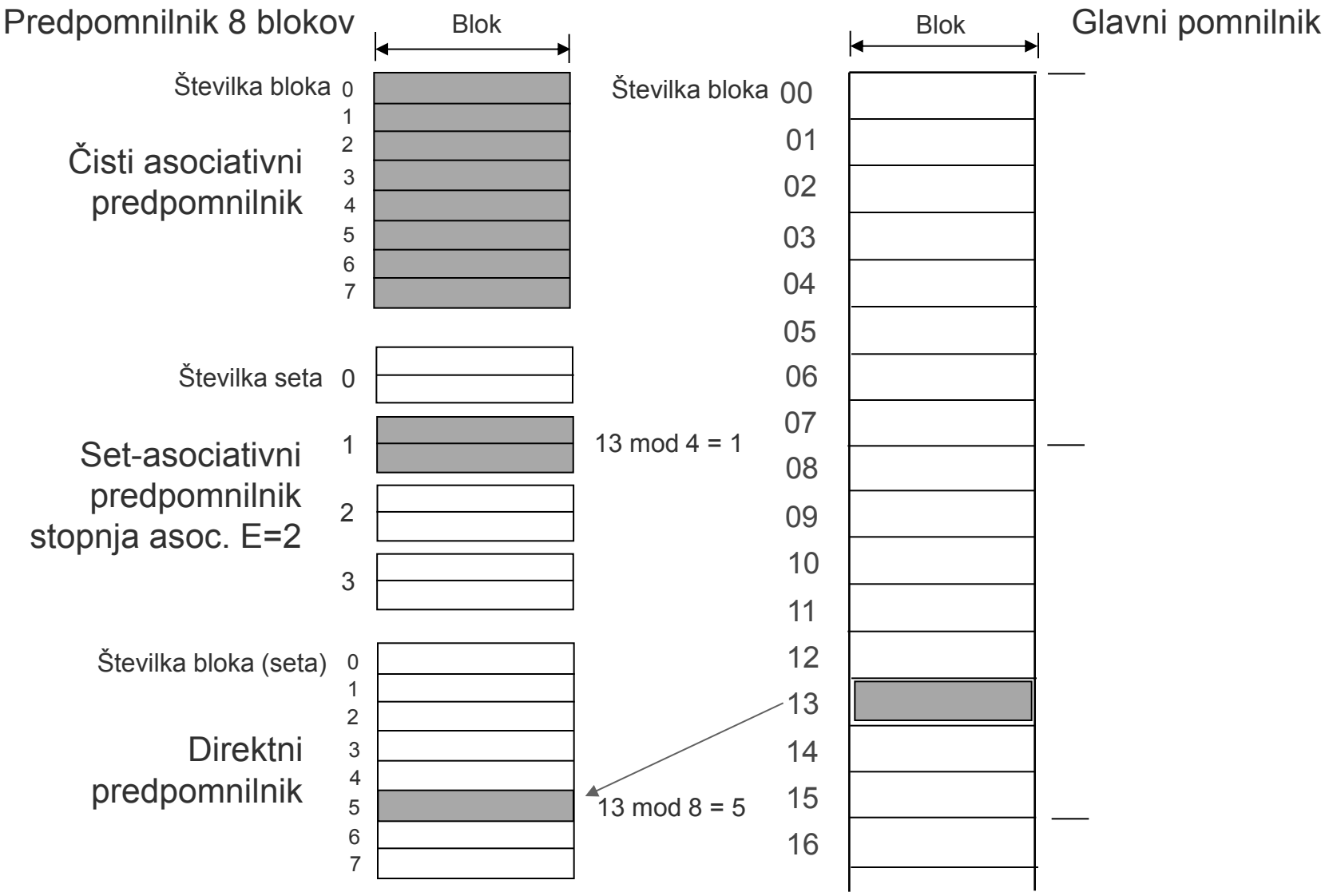
Določen blok iz glavnega pomnilnika se lahko preslika samo v točno določen blok predpomnilnika (vedno isti)

Pozicija v predpomnilniku =
(Štev.bloka) *mod* (Število blokov v predpomnilniku)









Vpliv predpomnilnika na hitrost delovanja CPE

- Dostop do predpomnilnika:
  - Zadetek:
    - Branje - običajno 1 urina perioda
    - Pisanje - branje bloka

      - spreminjanje vsebine

      - pisanje bloka nazaj - tipično ena urina perioda več

- Zgrešitev:
  - dostop do glavnega pomnilnika
  - prenos bloka do predpomnilnika
  - pisanje bloka v predpomnilnik
  - sledi branje ali pisanje kot pri zadetku
  - če je predpomnilnik poln, je potrebna še zamenjava bloka

- Za vse te operacije pri zgrešitvi je potrebnih od 10 do 100 urinih period (zgrešitvena kazen).
- Predpomnilniške zgrešitve zmanjšujejo hitrost delovanja CPE, oziroma povečujejo CPI.
- Idealni CPI (CPI_I) – brez upoštevanja zgrešitev v predpomnilniku
- Realni CPI (CPI_R) – z upoštevanjem zgrešitev v predpomnilniku

- Realni CPI z upoštevanjem zgrešitev v predpomnilniku je:

$$CPI_R = CPI_I + M_I(1-H) * Zgrešitvena\_kazen$$

CPI_R - realni CPI
CPI_I - idealni CPI brez zgrešitev v predpomnilniku
M_I - povprečno število pomn. dostopov na ukaz

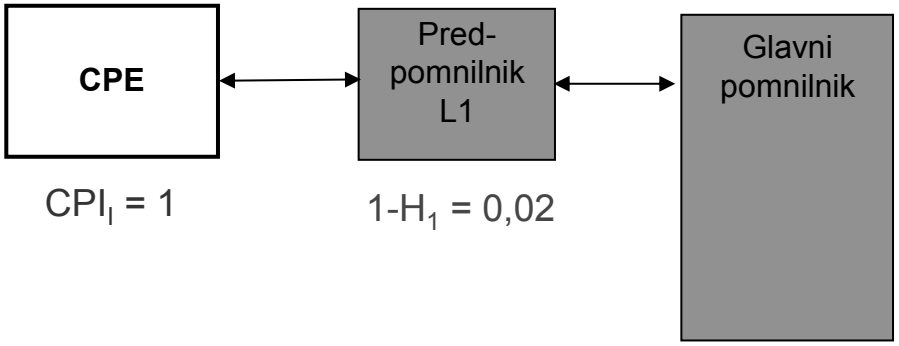
- Realni čas izvajanja programa z *N* ukazi pa je:

$$CPE_{čas} = N * CPI_R * t_{CPE}$$

Vpliv predpomnilnika L2 na hitrost CPE

- Procesor ima idealni $CPI_I = 1$, če v ukaznem predpomnilniku L1 ni zgrešitev
- Frekvenca ure procesorja $f_{CPE} = 4$ GHz
- Verjetnost zgrešitve v predpomnilniku L1 je 2%
- Zgrešitvena kazen je 100 ns (čas za prenos bloka iz glavnega pomnilnika)
- Če v hierarhijo dodamo še predpomnilnik L2 z zgrešitveno kaznijo 5 ns (čas za prenos bloka t_{B2}), je verjetnost zgrešitve v L2 0,5% (verjetnost, da bo potreben dostop do glavnega pomnilnika)
- Kolikokrat hitrejše je delovanje procesorja, če v hierarhijo dodamo predpomnilnik drugega nivoja L2?

Dvonivojska pomnilniška hierarhija



Zgrešitvena kazen t_{B2} (čas za prenos bloka iz glavnega pomnilnika v predpomnilnik)

t_{B2}= 100 ns = 400 [up]
[up] - urine periode

$$t_{CPE} = \frac{1}{f_{CPE}} = \frac{1}{4 \cdot 10^9}[s] = 0{,}25 \cdot 10^{-9}[s] = 0{,}25[ns]$$

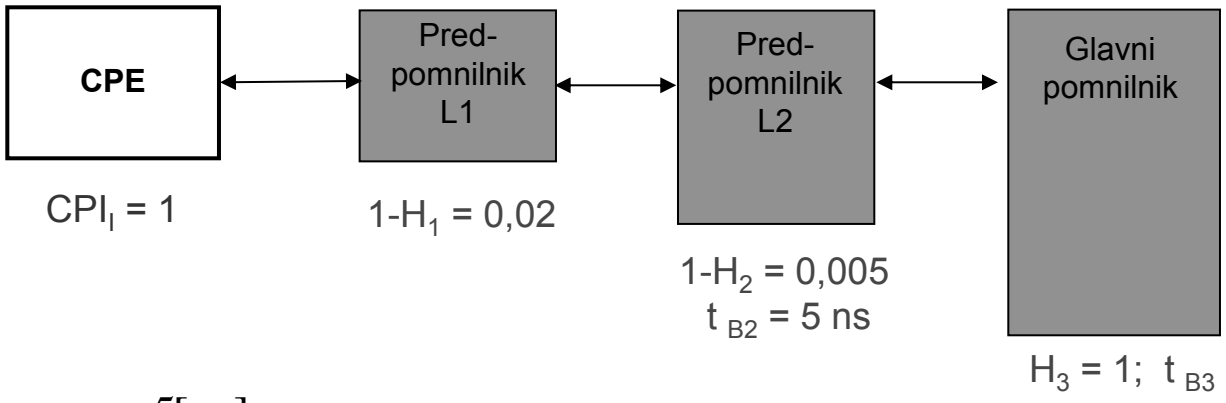
Čas trajanja ene urine periode

$$t_B = \frac{100[ns]}{0{,}25[\frac{ns}{up}]} = 400[up]$$

$$CPI_R(L1) = CPI_I + (1 - H_1) \cdot t_{B2} = 1[up] + 0{,}02 \cdot 400[up] = 9[up]$$

Zaradi zgrešitev v predpomnilniku se CPI iz 1 poveča na 9 urinih period

Trinivojska pomnilniška hierarhija



Zgrešitvena kazen t_{B3} (čas za prenos bloka iz glavnega pomnilnika v predpomnilnik)

t_{B3}= 100 ns = 400 [up]
[up] - urine periode

$$t_{B2} = \frac{5[ns]}{0{,}25[\frac{ns}{up}]} = 20[up]$$

Čas za prenos bloka iz L2 v L1

$$CPI_R(L1, L2) = CPI_I + (1 - H_1) \cdot t_{B2} + (1 - H_2) \cdot t_{B3} =$$
$$= 1[up] + 0{,}02 \cdot 20[up] + 0{,}005 \cdot 400[up] = 1 + 0{,}4 + 2 = 3{,}4[up]$$

$$Pohitritev = \frac{CPI_R(L1)}{CPI_R(L1, L2)} = \frac{9}{3{,}4} = 2{,}6$$

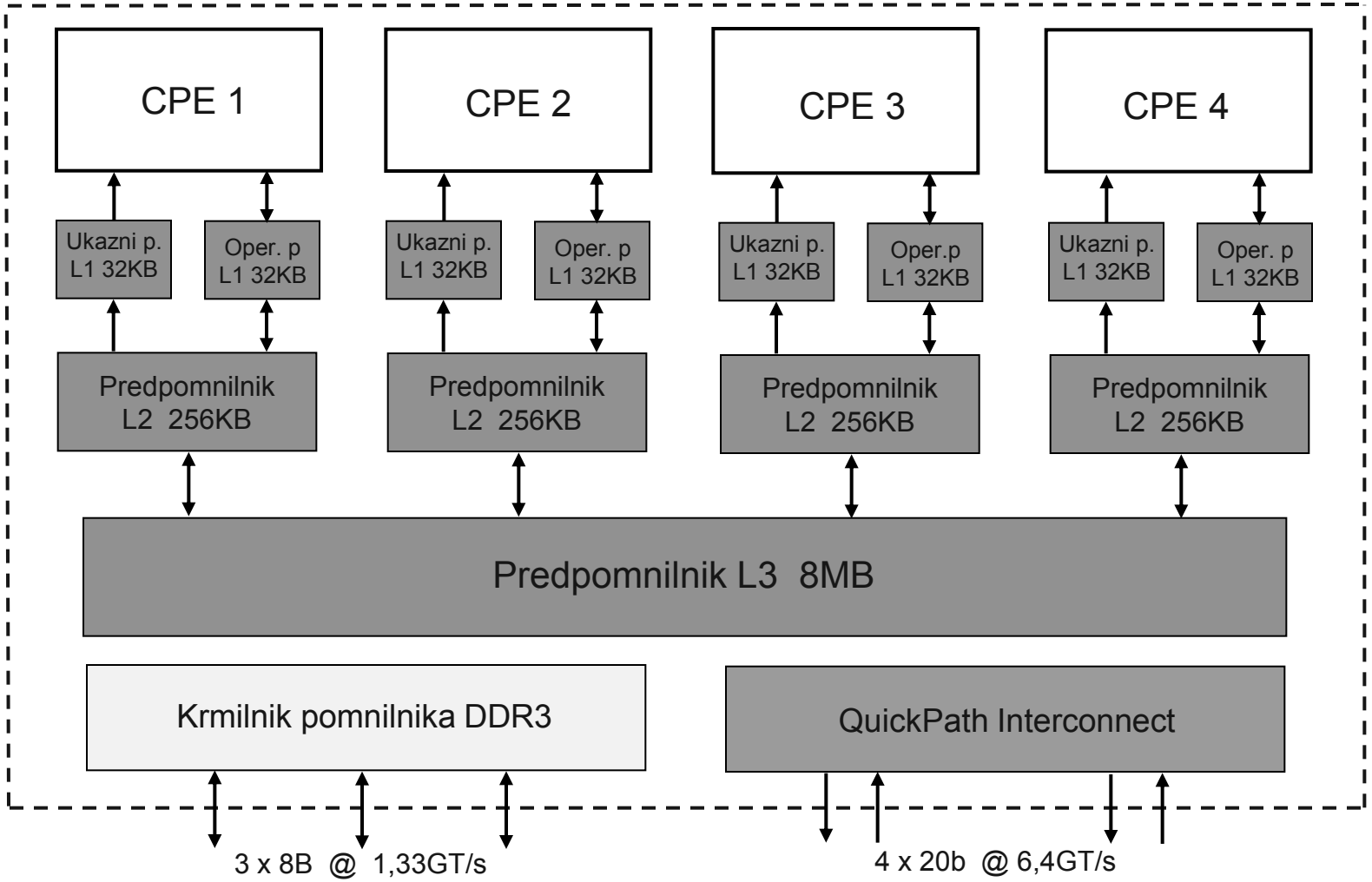
Če dodamo predpomnilnik L2, se hitrost izvajanja ukazov 2,6-krat poveča

Predpomnilniki pri nekaterih današnjih procesorjih

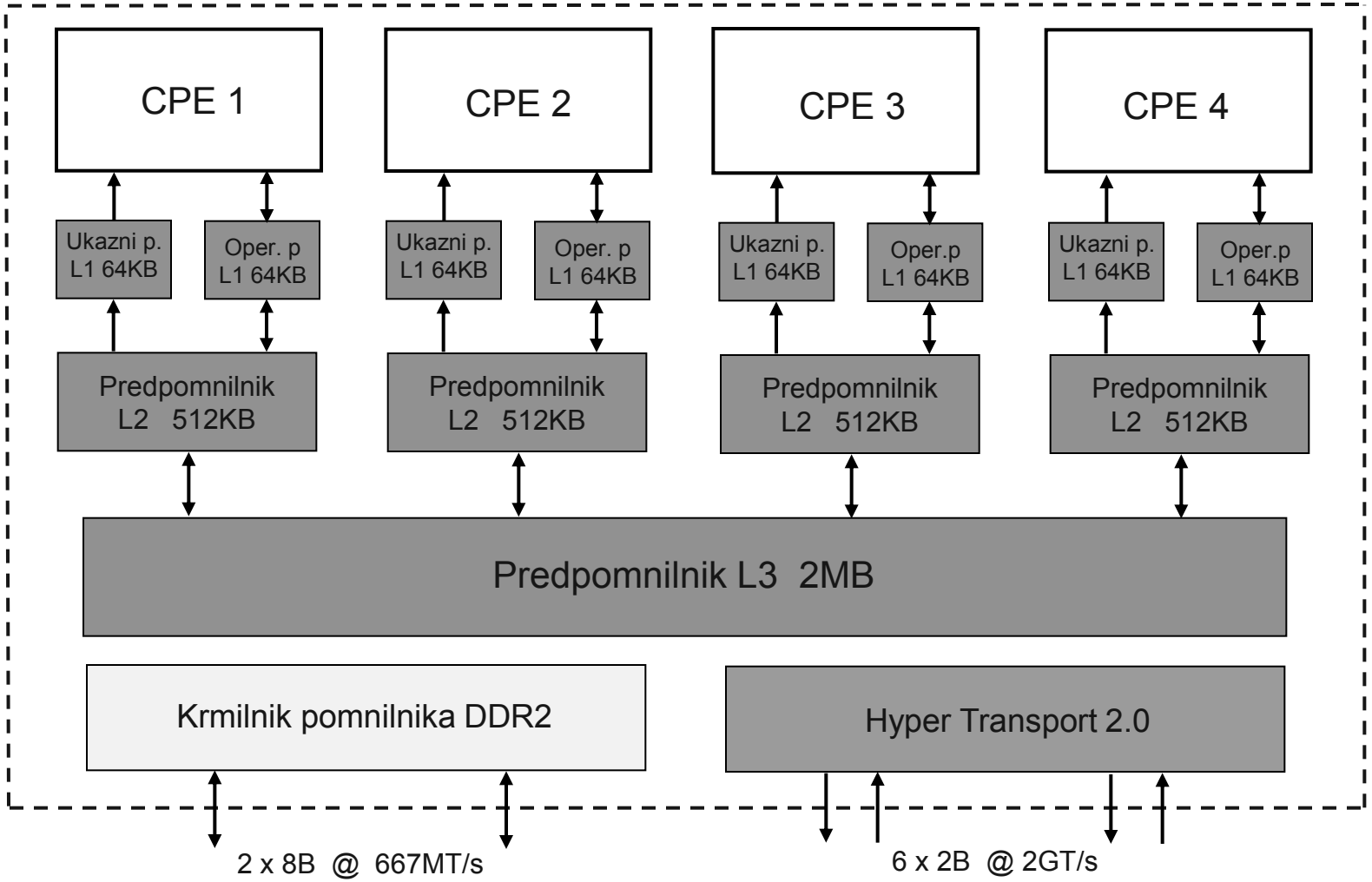
	Intel Core i7 Nehalem	AMD Opteron Barcelona	IBM Power6
Ukazni predpomnilnik L1			
Velikost	32KB	64KB	64KB
Stopnja asociativnosti	4	2	8
Velikost bloka	64B	64B	
Čas dostopa (urine periode – cikli)	4	3	4
Širina povezave do L2	256b	256b	
Operandni predpomnilnik L1			
Velikost	32KB	64KB	64KB
Stopnja asociativnosti	8	2	8
Velikost bloka	64B	64B	
Čas dostopa (urine periode - cikli)	4	3	4
Širina povezave do L2	256b	256b	

	Intel Core i7	AMD Opteron	IBM Power6
Predpomnilnik L2			
Velikost	256KB/jedro	512KB/jedro	4MB
Stopnja asociativnosti	8	16	
Velikost bloka	64B	64B	
Čas dostopa (urine periode – cikli)	<12		
Širina povezave do L2	256b	256b	
Predpomnilnik L3			izven CPE čipa
Velikost	8MB/štiri jedra	2MB/štiri jedra	32MB
Stopnja asociativnosti	16	32	
Velikost bloka	64B	64B	
Čas dostopa (urine periode – cikli)	30 - 40		
Širina povezave do glavnega pomn.	3 x 64b	2 x 64b	

Zgradba 4-jedrnega procesorja Intel Core i7 (Nehalem)



Zgradba 4-jedrnega procesorja AMD Opteron (Barcelona)



9.2 Navidezni pomnilnik

- Navidezni pomnilnik (virtual memory) $\Rightarrow$ prostor v pomožnem pomnilniku (SSD ali magnetni disk), ki je za uporabnika videti kot glavni pomnilnik.

- Dostop do pomožnega pomnilnika poteka z V/I ukazi, oziroma z V/I programi.

- Prenosi med glavnim in navideznim pomnilnikom so za uporabnika nevidni ($\Rightarrow$ navidezni pomnilnik)

- Potrebna je dodatna logika v CPE in programska oprema

- Navidezni pomnilnik ima danes večina računalnikov, razlog ni več samo premajhen glavni pomnilnik kot včasih, temveč tudi:
  - Veliko nižja cena pomnilnika na pomožnem pomnilniku.
  - Enostavna rešitev pozicijske neodvisnosti programov.
  - Zaščita pomnilnika.

- Prostor na pomožnem pomnilniku:
  - Prostor za navidezni pomnilnik.
  - Prostor za shranjevanje datotek (običajno precej večji).

- Čas dostopa in prenosa informacije (= zgrešitvena kazen) iz pomožnega pomnilnika v glavni pomnilnik je zelo dolg.

- Rešitve za zmanjšanje vpliva zelo velike zgrešitvene kazni pri navideznem pomnilniku:

  - Bloki morajo biti veliki (4KB, 8KB, do 64KB in več)

  - Vsak blok se lahko preslika v poljuben blok glavnega pomnilnika

  - Zamenjava blokov se ob zgrešitvah opravi programsko in ne strojno kot pri predpomnilniku

- Pomnilniški naslov iz CPE = **navidezni naslov** (ker se nanaša na navidezni pomnilnik).

- V povezavi z navideznim pomnilnikom imenujemo glavni pomnilnik **fizični pomnilnik.**

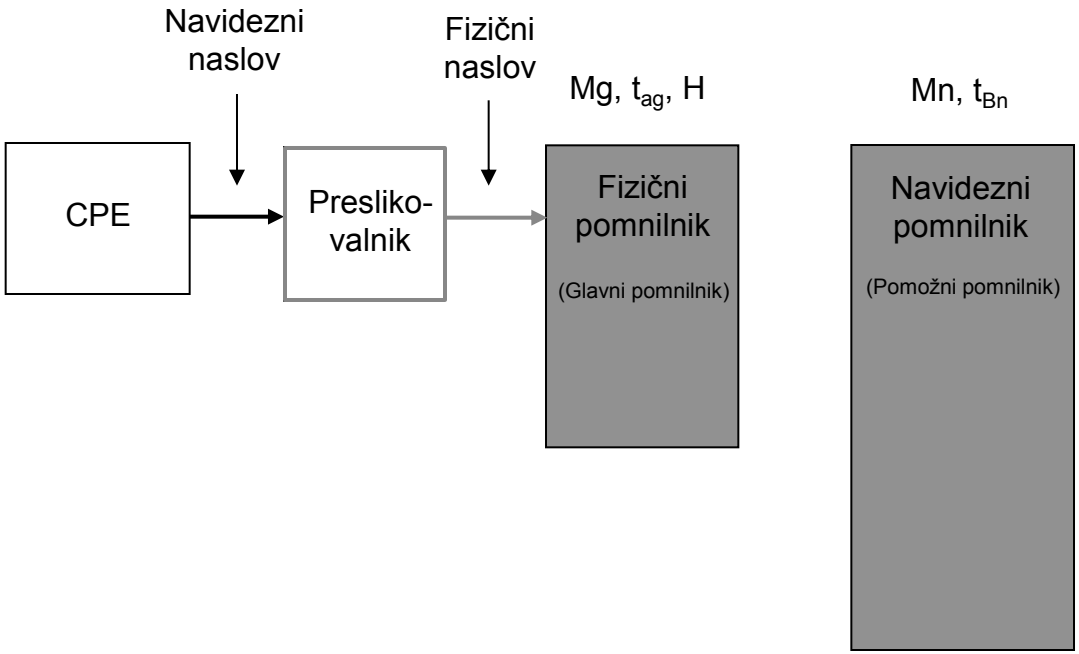
- Naslov, ki se nanaša na glavni pomnilnik = **fizični naslov**

- Pri vsakem pomnilniškem dostopu:
  Navidezni naslov → preslikava → fizični naslov

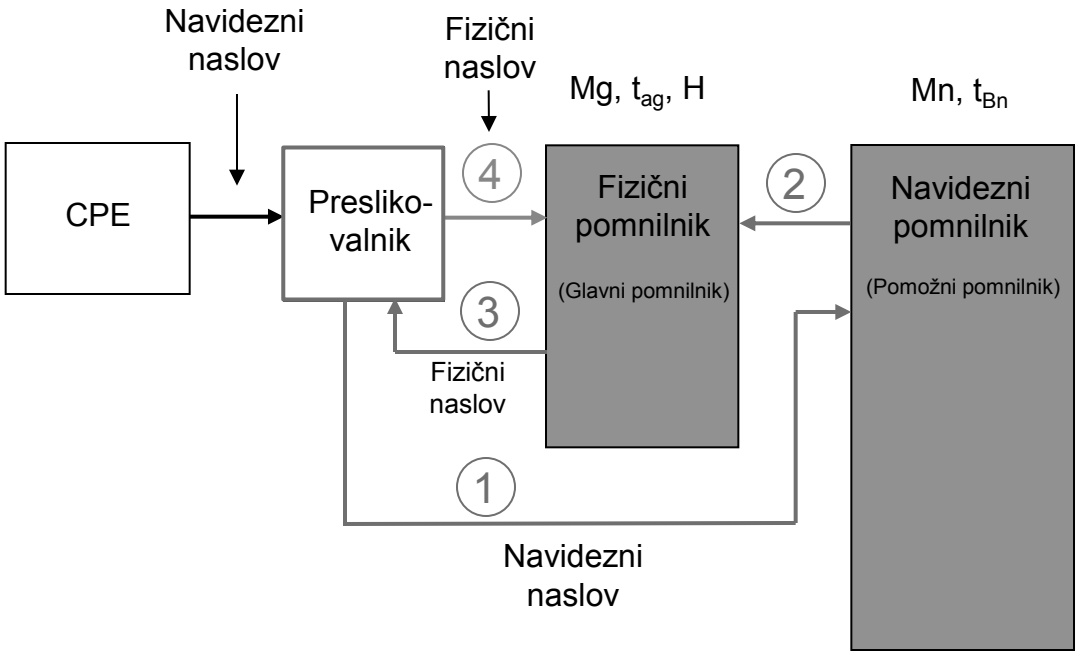
- Fizični naslov obstaja, če je v glavnem (fizičnem) pomnilniku zadetek.

- Pri večini računalnikov se fizični naslov (ne navidezni) uporabi za dostop do predpomnilnika.

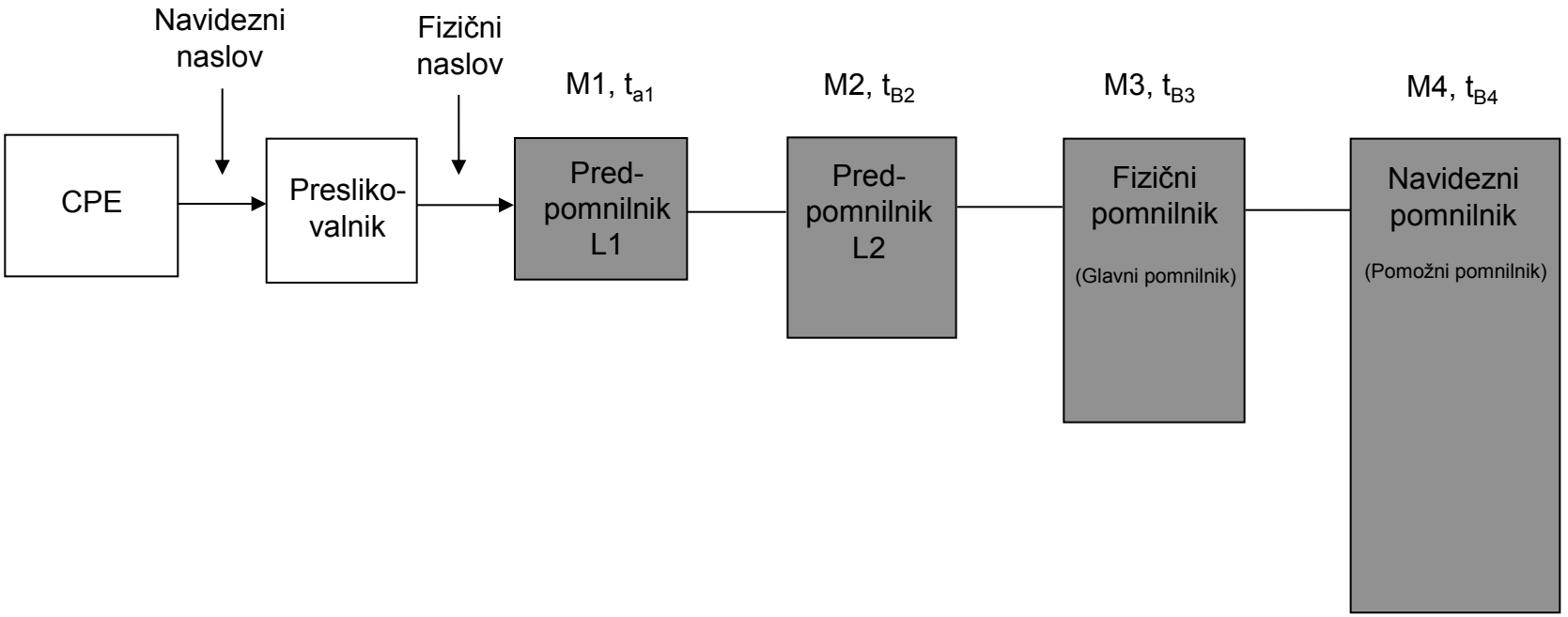
Naslovljena informacija je v fizičnem pomnilniku – zadetek
Verjetnost zadetka *H*



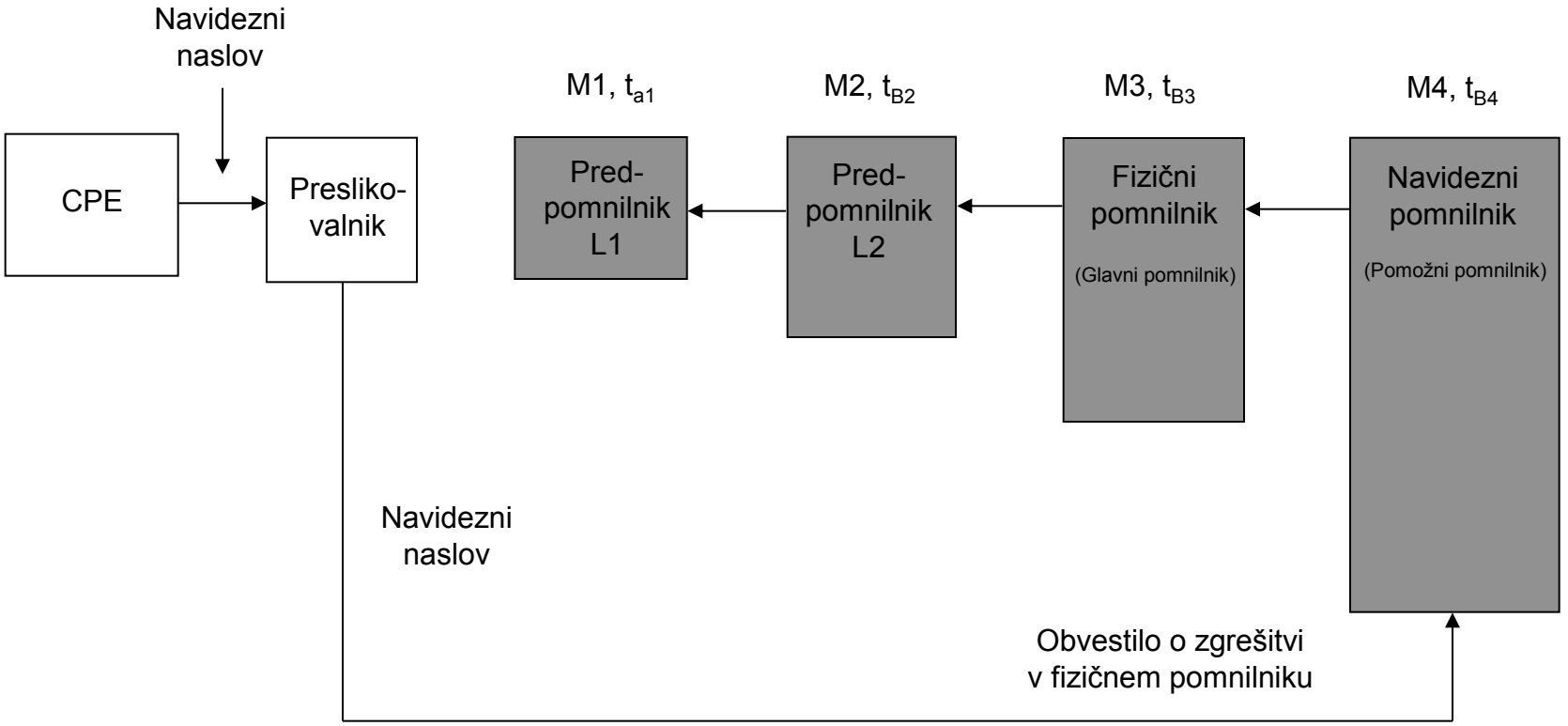
Naslovljena informacije ni v fizičnem pomnilniku – zgrešitev
Verjetnost zgrešitve *1-H*



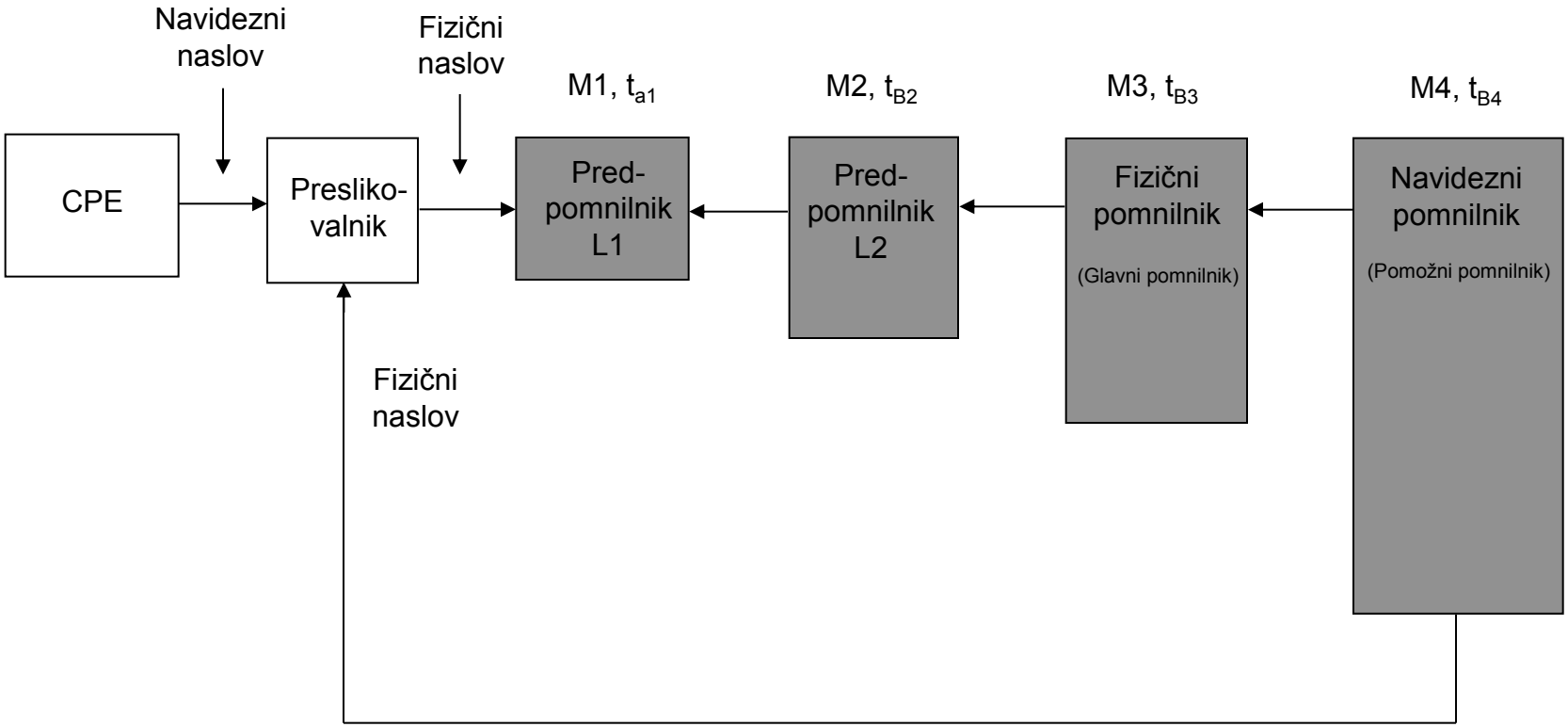
Celotna hierarhija
Naslovljena informacija je v fizičnem pomnilniku - zadetek



Celotna hierarhija
Naslovljene informacije ni v fizičnem pomnilniku - zgrešitev



Celotna hierarhija
Naslovljene informacije ni v fizičnem pomnilniku - zgrešitev



- Preslikovanje je možno tudi izklopiti $\Rightarrow$
  navidezni naslov = fizični naslov

- Preslikovalna funkcija se vzpostavi programsko (operacijski sistem)

- Pri vklopu računalnika je zato preslikovanje navideznih naslovov v fizične potrebno izklopiti (ker še ne deluje).

Navidezni pomnilnik z ostranjevanjem (paging)

- Pomožni pomnilnik ⇒ razdeljen na strani (pages):
  - strani ⇒ bloki enakih velikosti.
- Glavni pomnilnik ⇒ razdeljen na okvirje strani (page frames):
  - okvirji strani ⇒ enako veliki bloki kot v pomožnem pomnilniku.
- Število strani v navideznem pomnilniku je običajno veliko večje kot število okvirjev v glavnem pomnilniku:
  - iluzija o praktično neomejeno velikem pomnilniku.

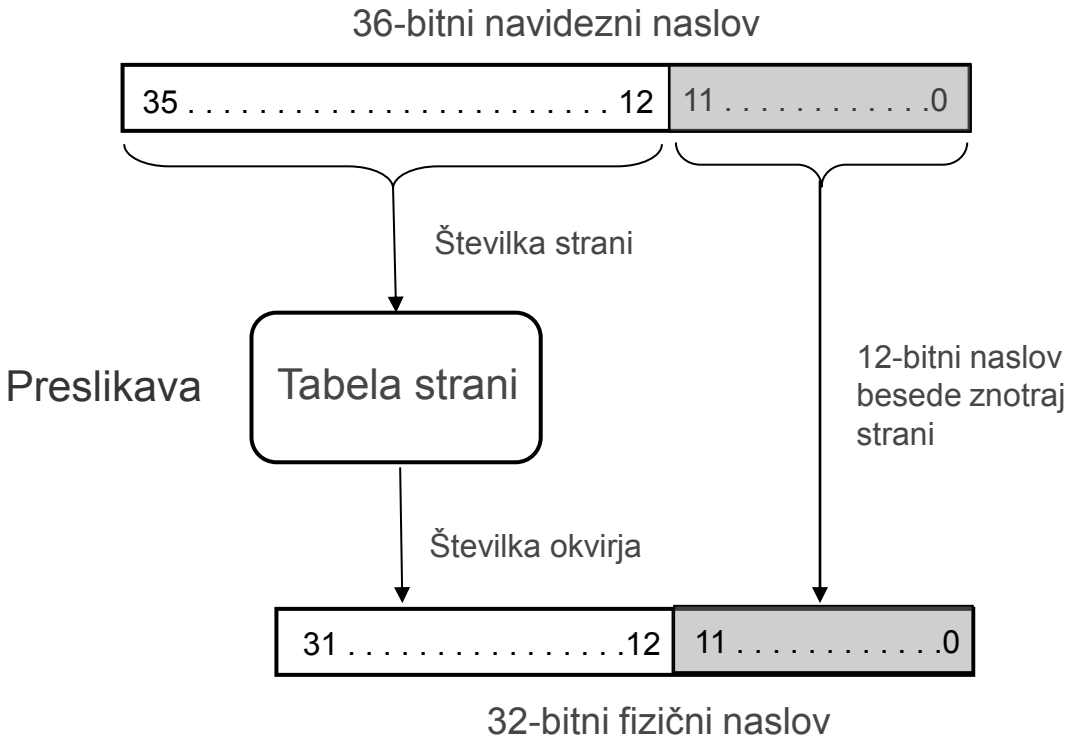
- Vsako stran iz navideznega je možno prenesti v poljuben okvir fizičnega pomnilnika.

- Za uporabnika je delitev pomnilniškega prostora na strani nevidna.

- Preslikava navideznega naslova (naslov strani) v fizični naslov (naslov okvirja) poteka preko tabele strani.

Primer preslikovanja navideznih naslovov v fizične pri ostranjevanju:

Velikost strani (in okvirja) 4 KB ($\Rightarrow 2^{12}$ B)
Navidezni naslov 36 bitov ($\Rightarrow$ Navidezni pomnilnik največ 2^{36} B = 64 GB)
Fizični naslov 32 bitov ($\Rightarrow$ Fizični pomnilnik največ 2^{32} B = 4 GB)



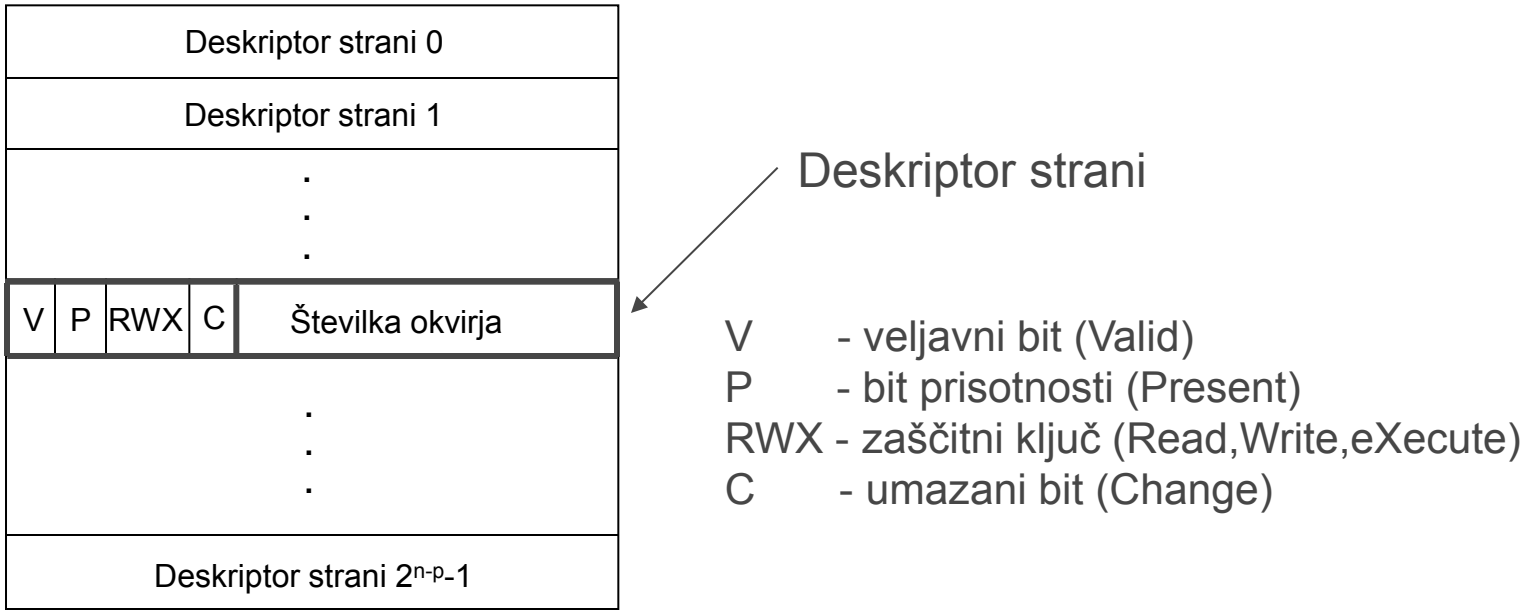
- Deskriptor strani (page descriptor) $\Rightarrow$ polje v tabeli strani, ki opisuje določeno stran.

- Število deskriptorjev v tabeli je enako številu strani v navideznem pomnilniku.

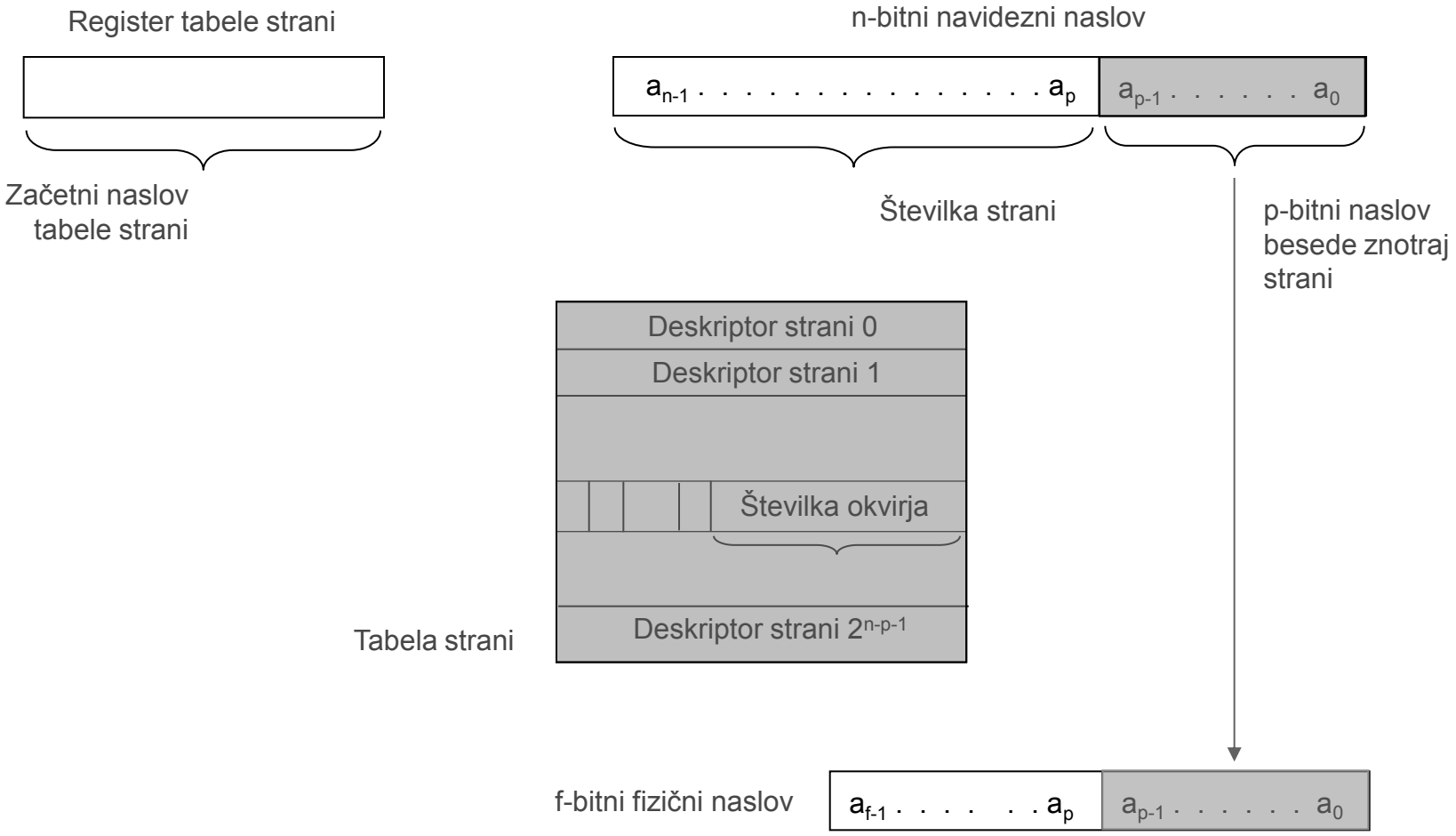
- Tabela strani je običajno v glavnem pomnilniku.

Zgradba tabele strani

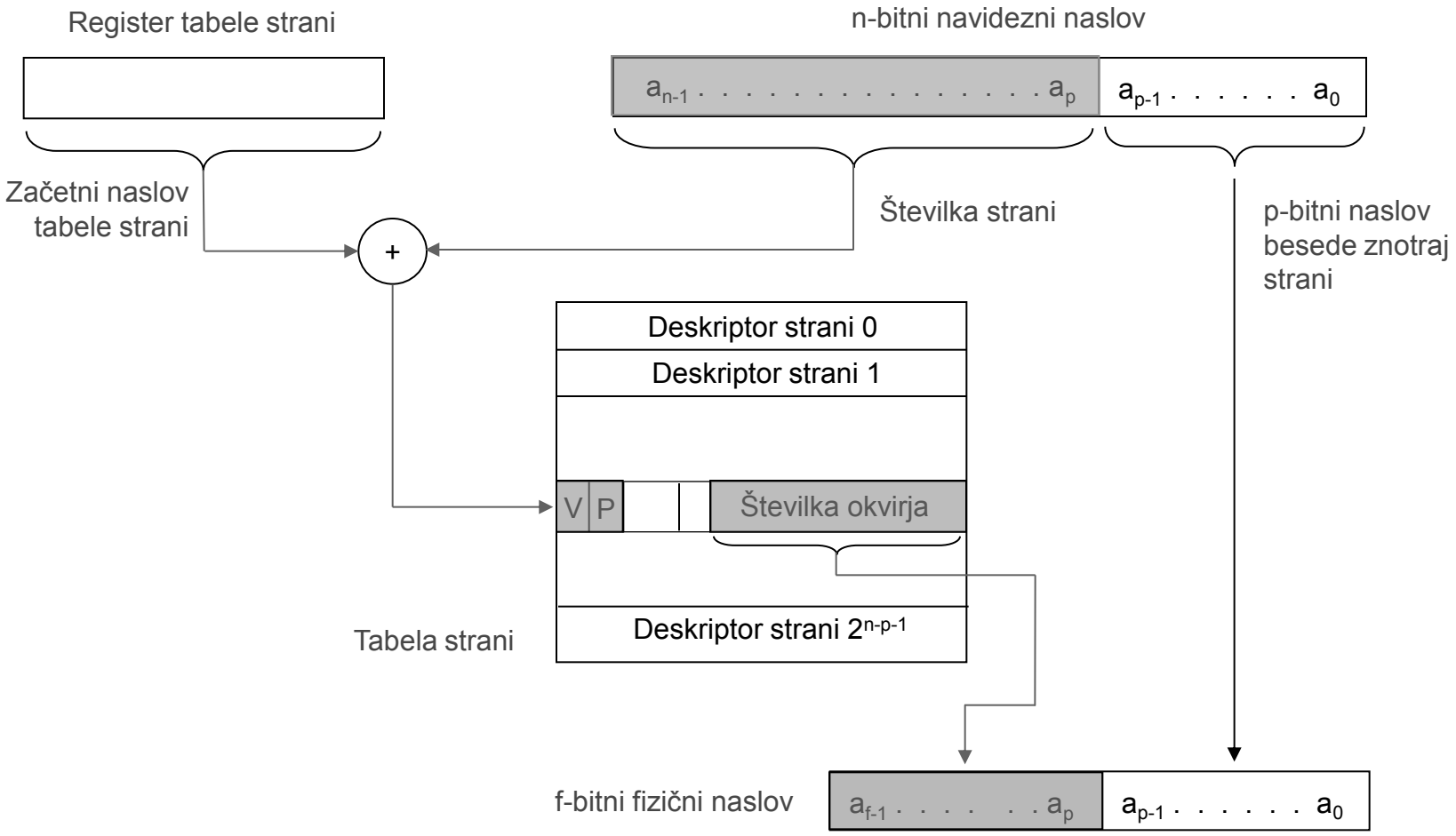
Velikost navideznega pomnilnika 2^n Bajtov (pri n=36 $\Rightarrow$ navidezni pomnilnik = 64 GB
Velikost strani 2^p Bajtov (pri p=12 $\Rightarrow$ velikost strani = 4 KB
Število strani v nav. pomnilniku = 2^{n-p} ($2^{36-12} = 2^{24}$ = 16 M strani (M = 2^{20}))
Število deskriptorjev v tabeli = število strani = 16 M



Preslikovanje navideznih naslovov v fizične pri ostranjevanju



Preslikovanje navideznih naslovov v fizične pri ostranjevanju



- Linearno preslikovanje – navidezni naslovni prostor je linearen. Pri računanju z navideznimi naslovi ni nobenih omejitev, kot če ne bi imeli navideznega pomnilnika.
- Delitev pomnilniškega prostora na strani je za uporabnika nevidna – običajnim programerjem sploh ni treba vedeti za obstoj strani.
- Tabela strani je običajno v glavnem pomnilniku
- Ena sama tabela strani $\Rightarrow$ Enonivojska preslikava

- Vsak program, ki se izvaja, ima svojo tabelo strani
- Če je velikost tabele strani npr. 32MB $\Rightarrow$ za vsak program (proces), ki se izvaja, 32MB pomnilnika za tabelo strani
- Stanje programa je določeno s tabelo strani, programskim števcem in registri (= proces)
- S tabelo strani je določen naslovni prostor, ki ga določen proces lahko uporablja.

- Tabele strani očitno v pomnilniku zasedejo veliko prostora
- Tabelo strani razdelimo na več nivojev $\Rightarrow$ več-nivojska preslikava
- Prednost: zmanjša se prostor, ki ga zasedajo tabele strani v glavnem pomnilniku.
- Največkrat se uporablja dve ali trinivojska preslikava preko dveh ali treh nivojev tabel strani.

- Operacijski sistem dodeljuje glavni (fizični) pomnilnik procesom in skrbi za ažuriranje tabel strani

- Navidezni pomnilnik omogoča uporabo glavnega pomnilnika večim procesom tako, da je:

  - pomnilniški prostor enega procesa zaščiten pred drugimi procesi

Napake strani

- Napaka strani (page fault): če navidezne strani ni v nobenem od okvirjev v glavnem pomnilniku (P-bit deskriptorja strani = 0), se sproži past za napako strani.

- Past za napako strani $\Rightarrow$ izvajati se začne servisni program, ki:
  - poišče stran v navideznem pomnilniku,
  - določi v kateri okvir v glavnem pomnilniku se bo stran preslikala in jo prenese,
  - ažurira deskriptor te strani v tabeli strani

- Ko operacijski sistem kreira proces, na disku običajno ustvari prostor za vse strani procesa (Swap space)

- Istočasno ustvari tudi podatkovno strukturo, ki za vsako stran vsebuje informacijo, kje je shranjena na disku.

Primerjava lastnosti predpomnilnika in navideznega pomnilnika z ostranjevanjem

	Predpomnilnik	Navidezni pomnilnik
Dostop	Predpomnilniška vrstica (blok)	Stran (okvir strani)
Blok	16B do 128B	4KB do 16KB (lahko tudi nekaj MB)
Verjetnost zgrešitve (1-H)	0,1% do 10% za L1	< 0,0001% (za glavni pomn.)
Zadetek	nekaj urinih period	~ 10 do 100 urinih period
Zgrešitvena kazen	~ 10 do 100 urinih ciklov	~ 10M urinih ciklov
Zamenjava bloka	Strojno (hardware)	Programsko (software)

Primerjava realizacije navideznega pomnilnika

	Intel Core i7 Nehalem	ARM Cortex-A8
Navidezni naslov	48 bitov	32 bitov
Fizični naslov	44 bitov	32 bitov
Velikost strani	4 KB, 2 MB, 4 MB	4, 16, 64 KB; 1, 16 MB

Pohitritev preslikovanja

- Pri preslikavi navideznega naslova v fizični naslov
  - je potreben dostop do tabele strani
  - tabele so shranjene v glavnem pomnilniku ali celo v navideznem pomnilniku
- Pri vsakem dostopu do pomnilnika sta zato potrebna dva dostopa do glavnega pomnilnika (če je preslikava enonivojska):
  - 1. dostop do deskriptorja v tabeli strani v glavnem pomnilniku
  - 2. dostop do želene besede na fizičnem naslovu v glavnem pomnilniku

- Pri več-nivojski preslikavi se poveča na 3 do 4 dostope do glavnega pomnilnika.
- Prepočasno!
- Rešitev: Predpomnilnik v CPE, ki vsebuje nekaj nazadnje uporabljenih deskriptorjev (nikoli operandov ali ukazov)

- **Preslikovalni predpomnilnik** (translation cache) = TLB
- **TLB** (Translation Lookaside Buffer)
- Dolžina bloka v preslikovalnem predpomnilniku je enaka dolžini deskriptorja, v kontrolnem delu pa je številka strani, ki ji deskriptor pripada.
- Visoko verjetnost zadetka (99% do 99,9%) se lahko doseže že z nekaj deskriptorji, zato je preslikovalni predpomnilnik lahko majhen in je večkrat čisti asociativni.

- Pri zadetku v preslikovalnem predpomnilniku dostop do tabele v glavnem pomnilniku ni potreben

- Pri Harvardski arhitekturi (ukazni in operandni predpomnilnik), sta potrebna tudi dva preslikovalna predpomnilnika (ukazni in operandni – ITLB in DTLB)

	Intel Core i7 Nehalem	AMD Opteron Barcelona
Navidezni naslov	48 bitov	48 bitov
Fizični naslov	44 bitov	48 bitov
Velikost strani	4KB, 2MB, 4MB	4KB, 2MB, 4MB
Preslikovalni predpomnilniki (TLB) za L1	Po en TLB za ukazni in operandni L1 pp Ukazni TLB 128 deskr. Oper. TLB 64 deskr. E=4, LRU zam.strategija	Po en TLB za ukazni in operandni L1 pp Oba TLB velikosti 48 deskr. Čista asociativna LRU zamen.strategija
Preslikovalni predpomnilniki (TLB) za L2	Velikost 512 deskr. E=4 LRU zam. strategija	Velikost 512 deskr. E=4

Strategije in algoritmi

- Delovanje navideznega pomnilnika vodi operacijski sistem, s ciljem doseči največjo izkoriščenost računalnika.

- Kot velika izkoriščenost se večinoma smatra, da se dana množica programov izvrši v najkrajšem možnem času.

- Na izkoriščenost računalnika vpliva izbira pravil, ki določajo:
  - Koliko okvirov strani naj ima v glavnem pomnilniku nek program
  - Kdaj, katere in koliko strani naj se prenese iz pomožnega v glavni pomnilnik
  - Katere strani naj se prenesejo iz glavnega pomnilnika nazaj v pomožni pomnilnik

- Ta pravila se imenujejejo **dodeljevalna, polnilna in zamenjevalna strategija**.

- Pri navideznem pomnilniku so strategije realizirane programsko, pri predpomnilniku pa strojno.

- Vse tri strategije izvajajo algoritmi s skupnim imenom **upravljanje s pomnilnikom** (memory management).

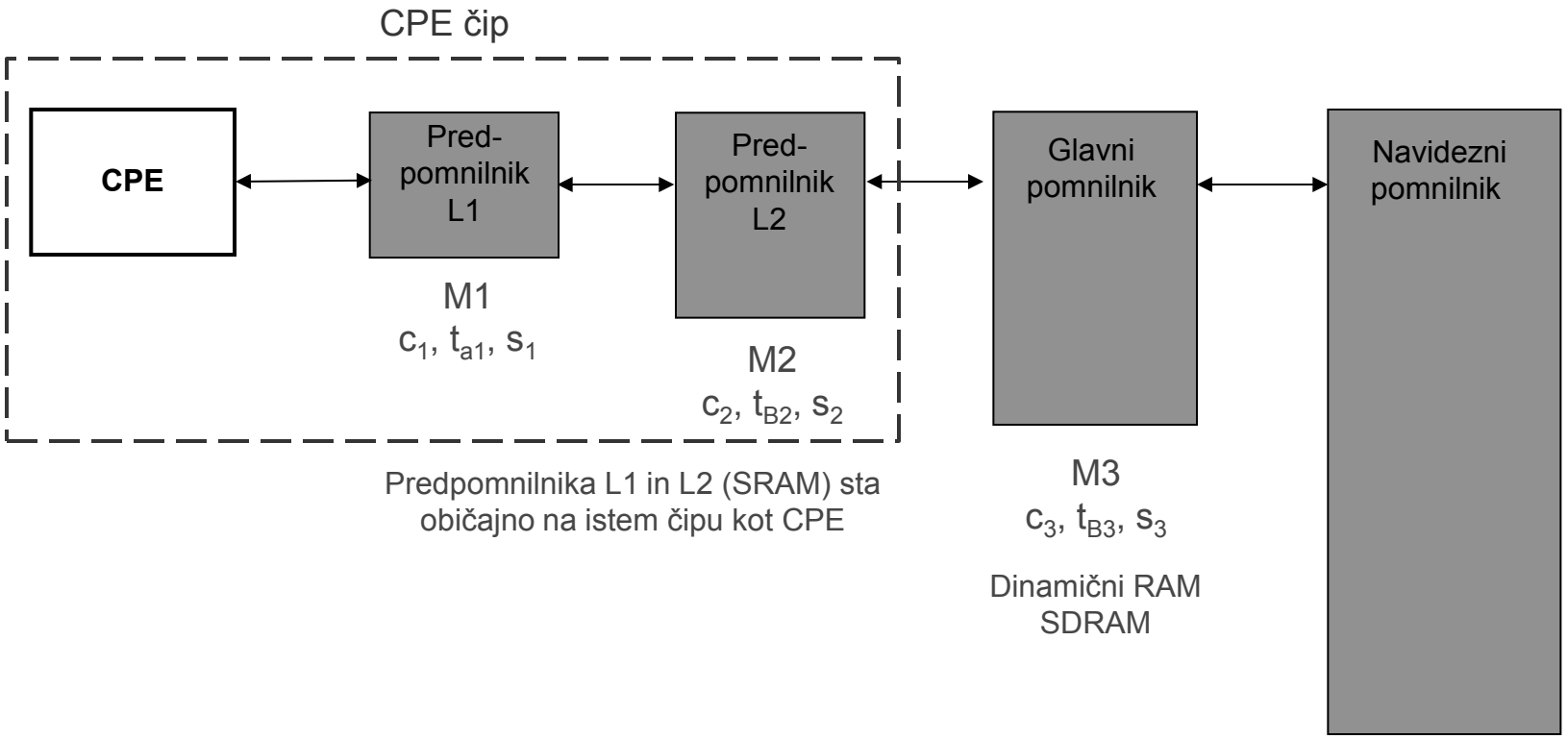
9.3 Delovanje pomnilniške hierarhije

- Pomnilniška hierarhija je iz CPE videti kot en sam pomnilnik:
  - S hitrostjo, ki je blizu hitrosti predpomnilnika (pomnilnika, ki je najbližji CPE)
  - Z velikostjo navideznega pomnilnika na pomožnem pomnilniku (zadnjega v hierarhiji)

- Pomnilniška hierarhija se od enonivojskega pomnilnika razlikuje v naslednjih lastnostih:
  - Čas dostopa ni enak za vse pomnilniške naslove, odvisen je od pomnilniškega nivoja na katerem je trenutno iskana pomnilniška beseda.
  - Za določen pomnilniški dostop ne moremo predvideti njegovega trajanja, znana je samo statistično določena povprečna vrednost časa dostopa.

- CPE pošlje v pomnilniško hierarhijo vedno naslov, ki se nanaša na pomnilnik M_n (zadnji v hierarhiji), vendar to ne pomeni, da bo dostop v resnici opravljen v M_n.
  - Če je informacija, do katere želi CPE dostop, v M_1 ($\Rightarrow$ zadetek), se opravi dostop do M_1
  - Če informacije ni v M_1 ($\Rightarrow$ zgrešitev), se v M_1 prenese iz M_2
  - Če želene informacije ni tudi v M_2, se v M_2 prenese iz M_3
  - . . .
  - Pri vsakem dostopu je zahtevana informacija vedno v pomnilniku M_n na zadnjem nivoju

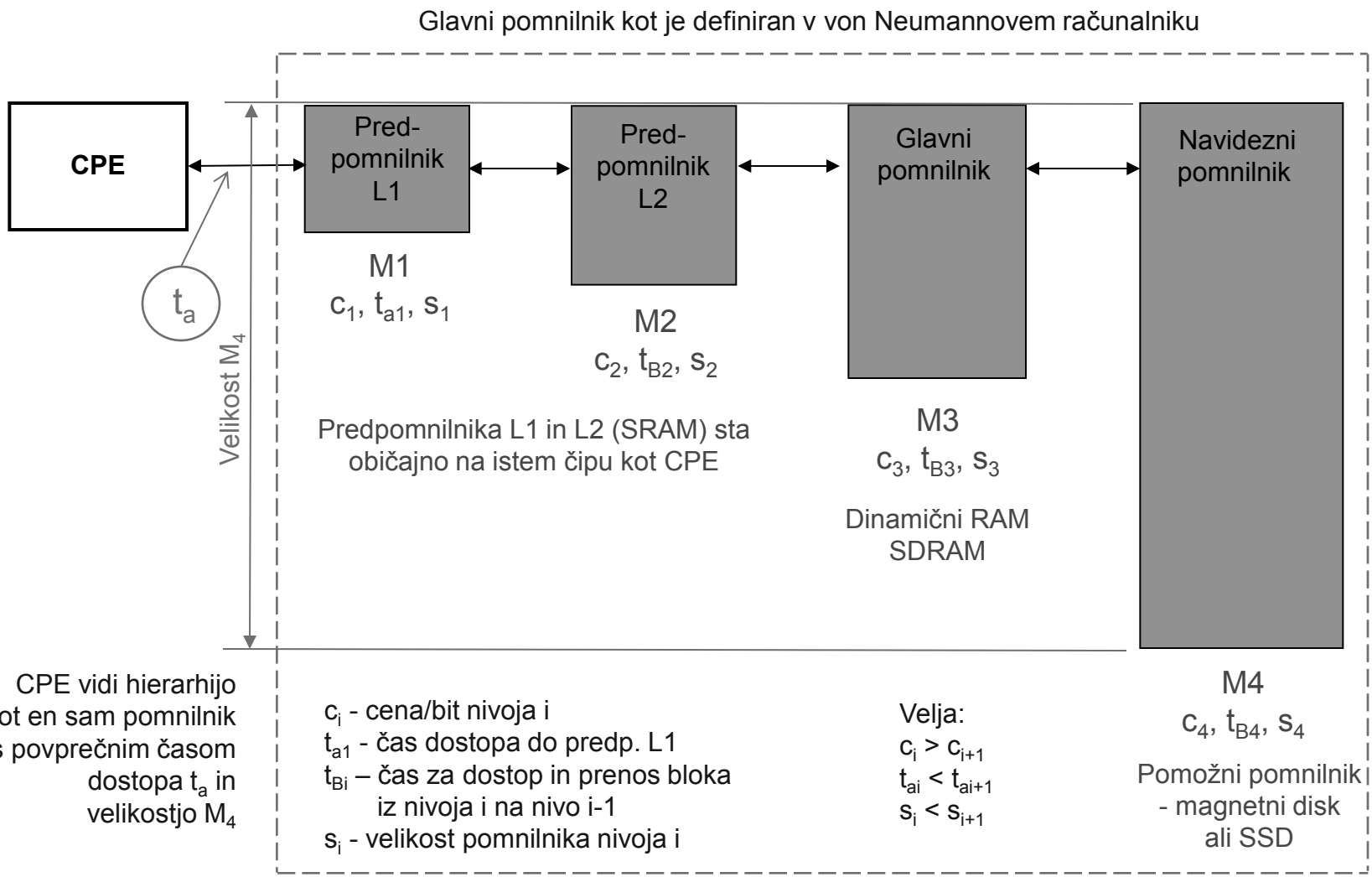
Štirinivojska pomnilniška hierarhija



c_i – cena/bit nivoja i
t_{a1} – čas dostopa do predpomnilnika L1
t_{Bi} – čas za dostop in prenos bloka iz nivoja i na nivo i-1
s_i – velikost pomnilnika nivoja i

Velja:
$c_i > c_{i+1}$
$t_{ai} < t_{ai+1}$
$s_i < s_{i+1}$

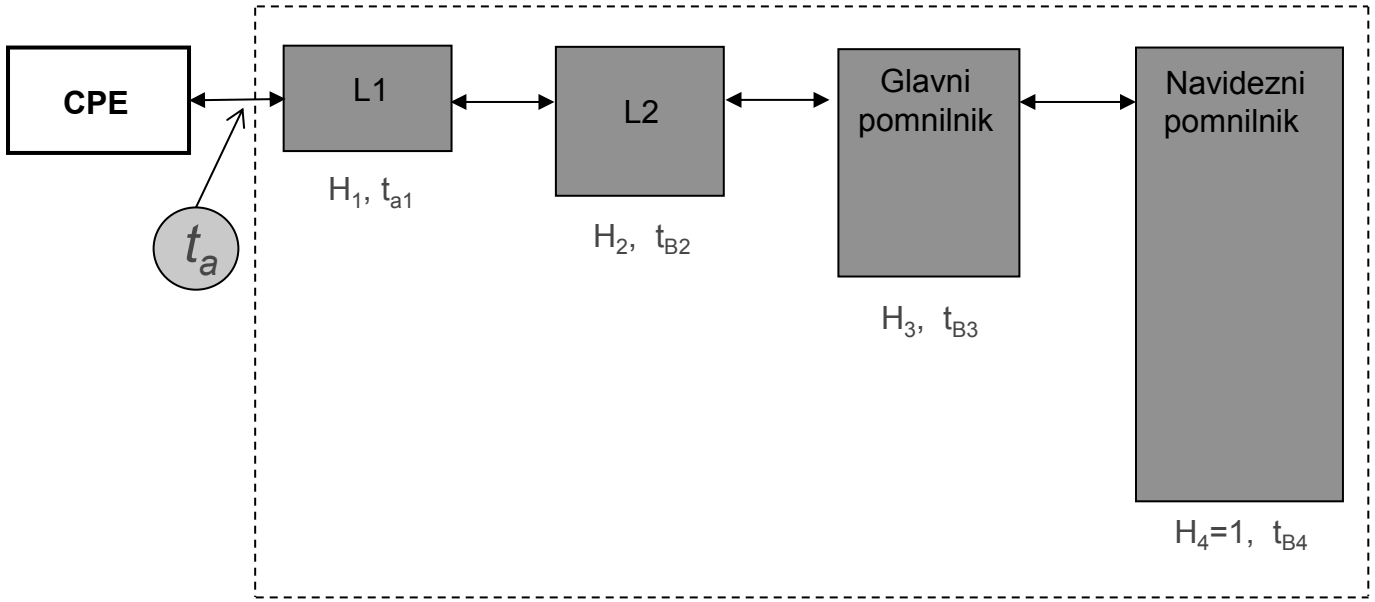
Štirinivojska pomnilniška hierarhija



- $H_i \Rightarrow$ verjetnost, da je pri poljubnem dostopu do pomnilniške hierarhije informacija na nivoju *i* in ne na nivoju *(i – 1).*
- $(1 - H_i) \Rightarrow$ verjetnost, da pri poljubnem dostopu želene informacije ni na nivoju *i*.
- Če je informacija na nivoju *i,* je zagotovo tudi na (*i+1).*
- Povprečni čas dostopa t_a do *n*-nivojske pomnilniške hierarhije, kot ga vidi CPE, je:

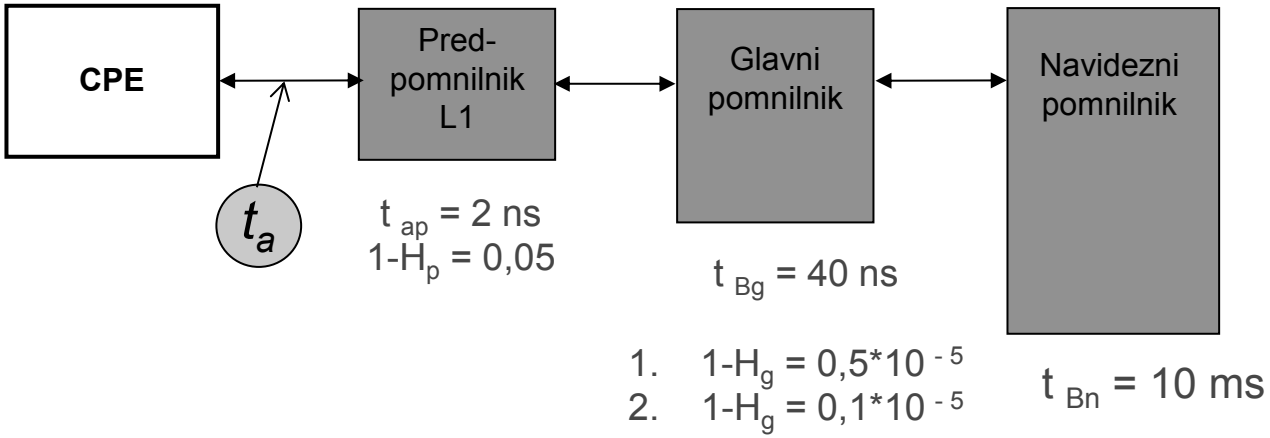
$$t_a = t_{a1} + (1 - H_1)t_{B2} + \ldots + (1 - H_{i-1})t_{Bi} + \ldots + (1 - H_{n-1})t_{Bn}$$

- Za štirinivojsko pomnilniško hierarhijo velja:



$$t_a = t_{a1} + (1 - H_1)t_{B2} + (1 - H_2)t_{B3} + (1 - H_3)t_{B4}$$

Primer vpliva verjetnosti zgrešitve v glavnem pomnilniku na povprečni dostopni čas pri trinivojski pomnilniški hierarhiji



t_{ap} – čas dostopa do predpomnilnika L1
H_p – verjetnost zadetka v predpomnilniku L1 ($1-H_p$ – verjetnost zgrešitve v L1)
t_{Bg} – čas za dostop do glavnega pomnilnika in prenos bloka iz glavnega pomnilnika v L1
H_g – verjetnost zadetka v glavnem pomnilniku ($1-H_g$ – verjetnost zgrešitve v glavnem pomnilniku)
t_{Bn} – čas za dostop do navideznega pomnilnika in prenos bloka iz navideznega pomnilnika v glavni pomnilnik
t_a – povprečni dostopni čas do celotne hierarhije, kot ga vidi CPE

1. Naj bo verjetnost zadetka v glavnem pomnilniku $H_g = 0{,}999995 = 99{,}9995\%$

   torej je verjetnost zgrešitve v glavnem pomnilniku $1\text{-}H_g = 0{,}000005 = 0{,}0005\%$
   ali $1\text{-}H_g = 0{,}5 * 10^{-5}$

   $t_{ap} = 2$ ns; $1\text{-}H_p = 0{,}05$; $t_{Bg} = 40$ ns; $t_{Bn} = 10$ ms

$$t_a = t_{ap} + (1 - H_p) \cdot t_{Bg} + (1 - H_g) \cdot t_{Bn} =$$
$$= 2 \cdot 10^{-9}[s] + 0{,}05 \cdot 40 \cdot 10^{-9}[s] + 0{,}5 \cdot 10^{-5} \cdot 10 \cdot 10^{-3}[s] =$$
$$= 2 \cdot 10^{-9}[s] + 2 \cdot 10^{-9}[s] + 5 \cdot 10^{-8}[s] =$$
$$= 2 \cdot 10^{-9}[s] + 2 \cdot 10^{-9}[s] + 50 \cdot 10^{-9}[s] = 54 \cdot 10^{-9}[ns] = 54[ns]$$

Povprečni dostopni čas v tem primeru je 54 ns, kar je slabše kot dostopni čas do glavnega pomnilnika (40 ns)

2. Če se verjetnost zadetka v glavnem pomnilniku poveča iz 99,9995% na 99,9999% $\Rightarrow H_g = 0{,}999999 = 99{,}9999\%$

   torej bo verjetnost zgrešitve v glavnem pomnilniku $1\text{-}H_g = 0{,}000001 = 0{,}0001\%$ ali $1\text{-}H_g = 0{,}1 * 10^{-5}$

   ostali podatki pa ostanejo nespremenjeni:
   $t_{ap} = 2$ ns; $1\text{-}H_p = 0{,}05$; $t_{Bg} = 40$ ns; $t_{Bn} = 10$ ms

2. Če se verjetnost zadetka v glavnem pomnilniku poveča iz 99,9995% na 99,9999% $\Rightarrow H_g = 0{,}999999 = 99{,}9999\%$

   torej bo verjetnost zgrešitve v glavnem pomnilniku $1\text{-}H_g = 0{,}000001 = 0{,}0001\%$ ali $1\text{-}H_g = 0{,}1 * 10^{-5}$

   ostali podatki pa ostanejo nespremenjeni:
   $t_{ap} = 2$ ns; $1\text{-}H_p = 0{,}05$; $t_{Bg} = 40$ ns; $t_{Bn} = 10$ ms

$$t_a = t_{ap} + (1 - H_p) \cdot t_{Bg} + (1 - H_g) \cdot t_{Bn} =$$
$$= 2 \cdot 10^{-9}[s] + 0{,}05 \cdot 40 \cdot 10^{-9}[s] + 0{,}1 \cdot 10^{-5} \cdot 10 \cdot 10^{-3}[s] =$$
$$= 2 \cdot 10^{-9}[s] + 2 \cdot 10^{-9}[s] + 1 \cdot 10^{-8}[s] =$$
$$= 2 \cdot 10^{-9}[s] + 2 \cdot 10^{-9}[s] + 10 \cdot 10^{-9}[s] = 14 \cdot 10^{-9}[ns] = 14[ns]$$

2. Če se verjetnost zadetka v glavnem pomnilniku poveča iz 99,9995% na 99,9999% $\Rightarrow H_g = 0{,}999999 = 99{,}9999\%$

   torej bo verjetnost zgrešitve v glavnem pomnilniku $1\text{-}H_g = 0{,}000001 = 0{,}0001\%$ ali $1\text{-}H_g = 0{,}1 * 10^{-5}$

   ostali podatki pa ostanejo nespremenjeni:
   $t_{ap} = 2$ ns; $1\text{-}H_p = 0{,}05$; $t_{Bg} = 40$ ns; $t_{Bn} = 10$ ms

$$t_a = t_{ap} + (1 - H_p) \cdot t_{Bg} + (1 - H_g) \cdot t_{Bn} =$$
$$= 2 \cdot 10^{-9}[s] + 0{,}05 \cdot 40 \cdot 10^{-9}[s] + 0{,}1 \cdot 10^{-5} \cdot 10 \cdot 10^{-3}[s] =$$
$$= 2 \cdot 10^{-9}[s] + 2 \cdot 10^{-9}[s] + 1 \cdot 10^{-8}[s] =$$
$$= 2 \cdot 10^{-9}[s] + 2 \cdot 10^{-9}[s] + 10 \cdot 10^{-9}[s] = 14 \cdot 10^{-9}[ns] = 14[ns]$$

Če se verjetnost zgrešitve v glavnem pomnilniku zmanjša iz $0{,}5 * 10^{-5}$ na $0{,}1 * 10^{-5}$ (verjetnost zadetka poveča), se povprečni dostopni čas zmanjša iz 54 ns na 14 ns.

