OSNOVE RAČUNALNIŠKE ARHITEKTURE II

10 Predpomnilnik

- Razkorak med zmogljivostjo CPE in hitrostjo DRAM pomnilnikov se stalno povečuje.
- Razliko med hitrostjo CPE in DRAMov lahko zmanjšamo z uporabo predpomnilnika, vendar samo to ni dovolj.
- Dodatni možnosti za pohitritev pri uporabi pred-pomnilnika sta:
  - Uporaba dveh predpomnilnikov enega za ukaze in drugega za operande (Harvardska arhitektura)
  - Uporaba širših podatkovnih poti med predpomnilnikom in CPE

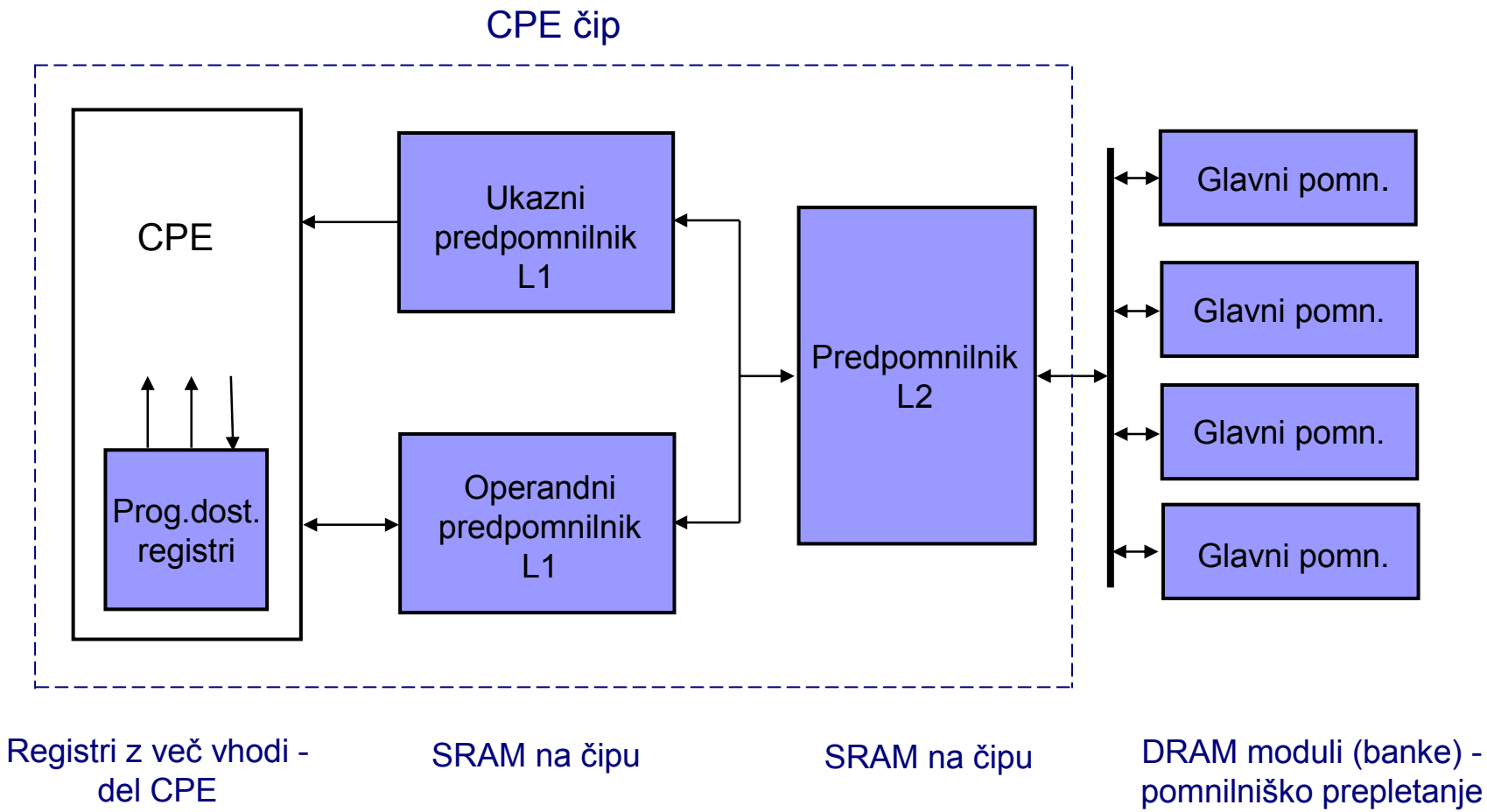
- Če iskane besede ni v predpomnilniku, je potreben dostop do glavnega pomnilnika (DRAM) in prenos novega bloka, ki vsebuje iskano besedo, v predpomnilnik.

- Zato je potrebno pohitriti tudi ta prenos, rešitve so:
  - Ker so besede v bloku sosednje, se lahko uporabi pri dostopu do DRAMa page mode način prenosa
  - Tudi pri prenosu med predpomnilnikom in DRAMom se lahko uporabi širše podatkovne poti
  - Razdelitev glavnega pomnilnika na več modulov - pomnilniško prepletanje

- Drugi predpomnilniški nivo L2 izven CPE, ki je vedno homogen (Princetonska arhitektura). Včasih tudi tretji nivo L3.
- S CPE je predpomnilnik L2 lahko povezan s posebno prenosno potjo. V tem primeru ima CPE dve prenosni poti:
  - Eno za povezavo z glavnim pomnilnikom
  - Drugo za povezavo s predpomnilnikom nivoja L2

- Te rešitve se lahko kombinirajo, čeprav se danes pri zmogljivejših računalnikih uporabljajo večinoma kar vse.

Predpomnilnik



Osnove delovanja predpomnilnikov

- Predpomnilnik (cache) je majhen hiter pomnilnik med CPE in glavnim pomnilnikom.
- Vsebina predpomnilnika so deli vsebine glavnega pomnilnika, oziroma podmnožica vsebine glavnega pomnilnika.
- Razlog, da bo iskana informacija dovolj pogosto v predpomnilniku je **lokalnost pomnilniških dostopov.**
- Predpomnilnik mora biti narejen tako, da vsebuje delovno množico *V(t,T)* in se prilagaja njenemu spreminjanju.

- Delovna množica *V(t,T)* je množica različnih pomnilniških naslovov, ki jih program generira v času od *t-T* do *T*.
- Če v času *T* pride v pomnilnik *N* pomnilniških naslovov, velja:
  - Velikost množice *V(t,T)* je veliko manjša od *N*
  - Vsebina zaporedno si sledečih množic se spreminja počasi
- Ker je delovna množica manjša od glavnega pomnilnika, je dovolj že majhen predpomnilnik.

- Ker je v predpomnilniku samo del vsebine glavnega pomnilnika, mora biti v predpomnilniku še informacija, kateri deli vsebine glavnega pomnilnika so trenutno v njem.

- Pri dostopu CPE do informacije (ukaz, operand) sta dve možnosti:
  - **Zadetek** (hit), če je naslov v predpomnilniku
  - **Zgrešitev** (miss), če naslova ni v predpomnilniku

- Uspešnost delovanja predpomnilnika merimo:

- Z verjetnostjo zadetka *H* $H = \frac{N_p}{N} = \frac{N_p}{N_g + N_p}$

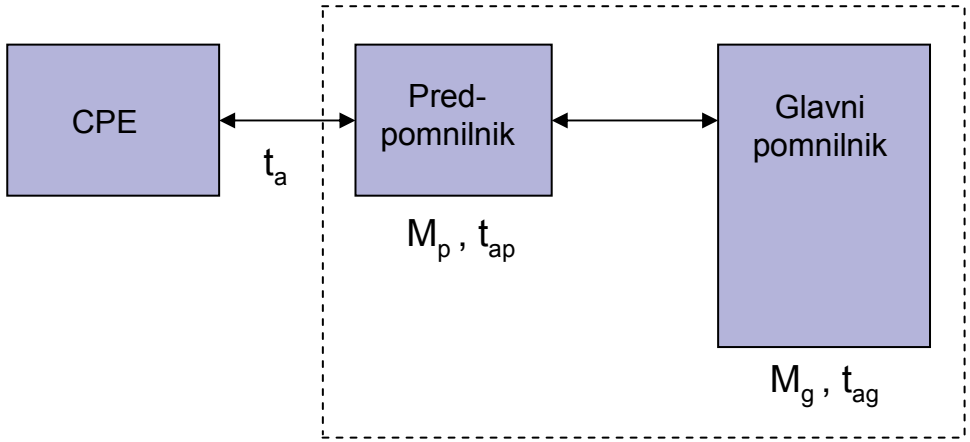
  N - število vseh dostopov ($N = N_g + N_p$)
  N_p - število zadetkov
  N_g - število zgrešitev

- Z verjetnostjo zgrešitve *1-H*

- **Verjetnost zadetka *H*** je pri današnjih predpomnilnikih večja od 0,9; večinoma celo večja od 0,95.

- To pomeni, da bo npr. pri 100 dostopih do pomnilnika pri 95 v predpomnilniku zadetek (vendar vnaprej ne vemo pri katerih), pri 5 dostopih pa zgrešitev.

- V primeru zgrešitve je potreben dostop do glavnega pomnilnika.
- Čas dostopa do predpomnilnika označimo s t_{ap}, *čas* dostopa do glavnega pomnilnika pa s t_{ag}.



Povprečen čas dostopa t_a do predpomnilnika in glavnega pomnilnika skupaj je:

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

- Pri računanju je treba upoštevati dve posebnosti predpomnilnikov:
  - Pri zgrešitvi se iz glavnega pomnilnika ne prenese samo ena beseda, temveč **predpomnilniški blok**, kar je več sosednjih besed
  - Čas v računalniku merimo z urinimi periodami
- Čas t_{ap} je število urinih period za dostop do predpomnilnika na nivoju L1 in je pri večini računalnikov 1 urina perioda.
- Pri zgrešitvi je potreben dostop do glavnega pomnilnika, čas ki je zato potreben, pa je **zgrešitvena kazen.**

- Zgrešitvena kazen t_B je število urinih period, ki se ob zgrešitvi prištejejo k času dostopa.
- Zgrešitvena kazen je odvisna od velikosti bloka, širine podatkovne poti med predpomnilnikom in DRAMom in je tipično med 10 in 100 urinimi periodami.
- Če ima računalnik tudi predpomnilnik na nivoju L2, je zgrešitvena kazen precej manjša.

- Povprečni čas dostopa t_a je potem:

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

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

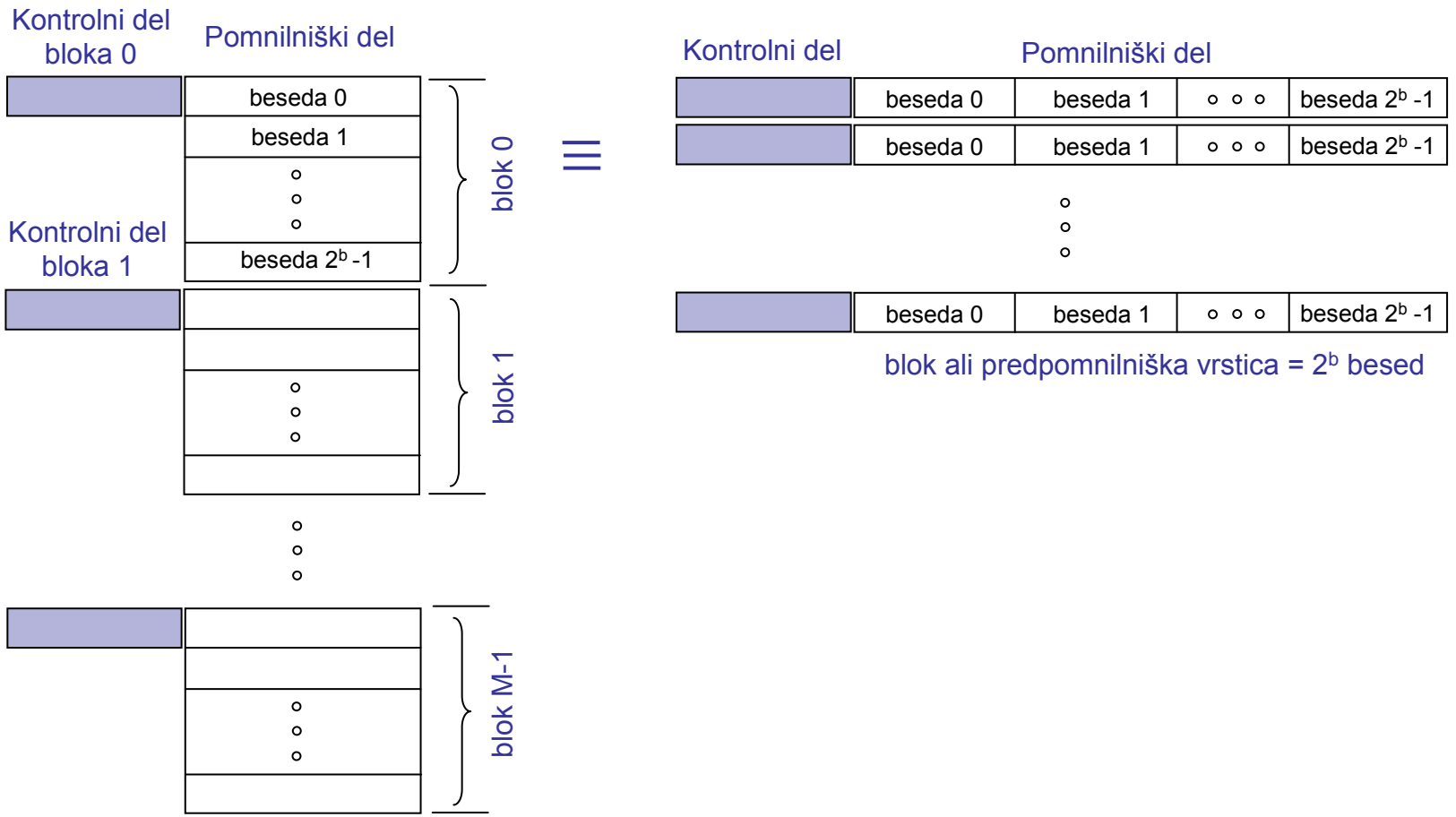
- Povprečni dostopni čas v sekundah pa lahko izračunamo tako, da rezultat t_a v urinih periodah pomnožimo s časom trajanja ene urine periode t_{CPE}.

- Ker se vsebina v predpomnilniku spreminja, mora predpomnilnik vsebovati poleg besed iz glavnega pomnilnika tudi naslove teh besed. (Kopija katerih besed iz glavnega pomnilnika je v predpomnilniku).

- Zato je vsak predpomnilnik sestavljen iz dveh delov:
  - Kontrolnega dela
  - Pomnilniškega dela

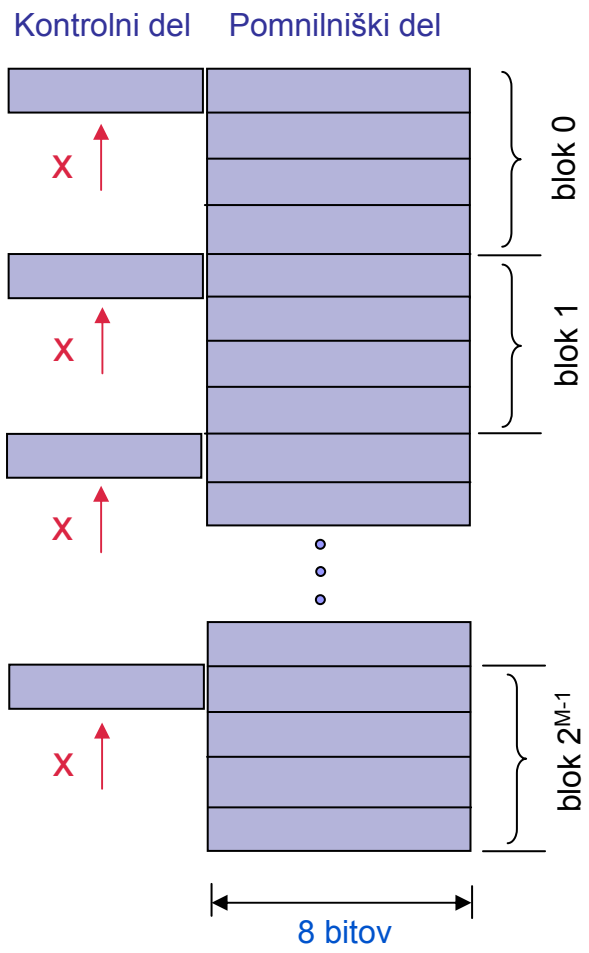
- Pomnilniški del je razdeljen na enake dele, **bloke** ali **predpomnilniške vrstice**.

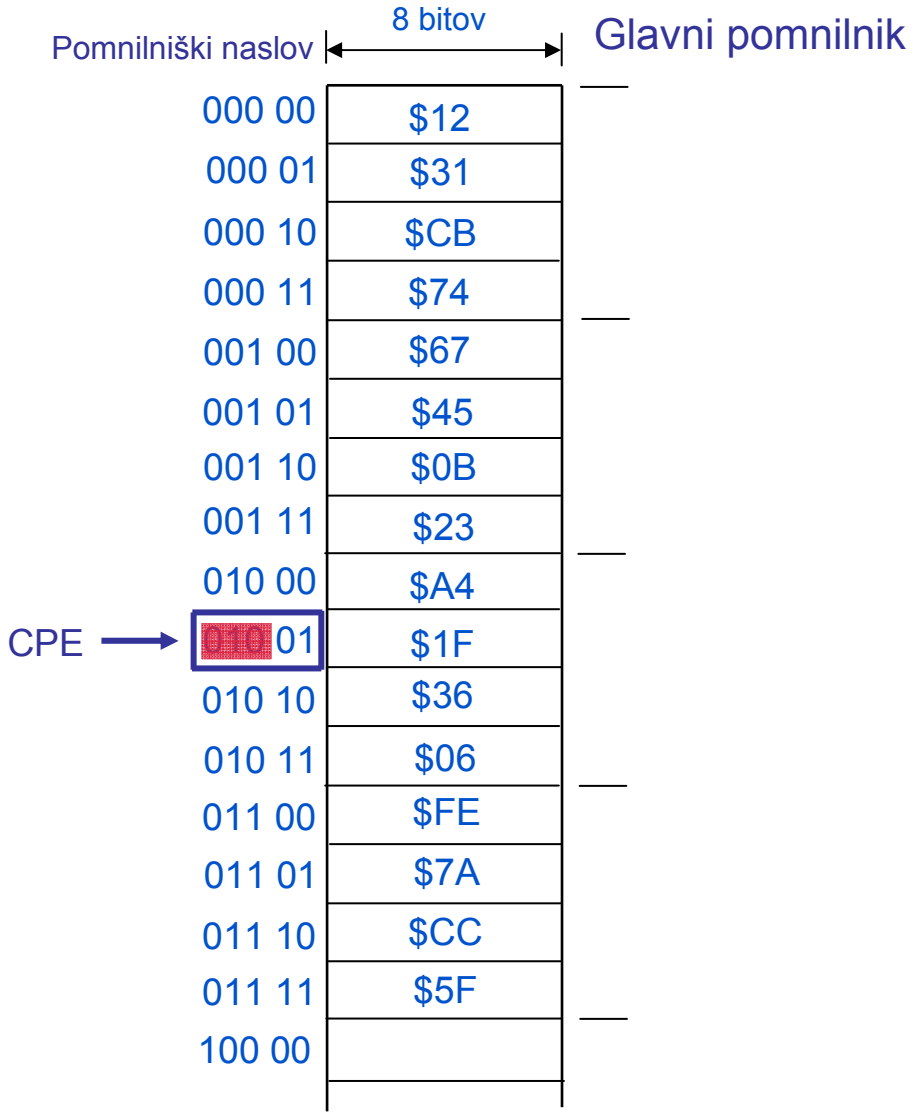
Zgradba predpomnilnika



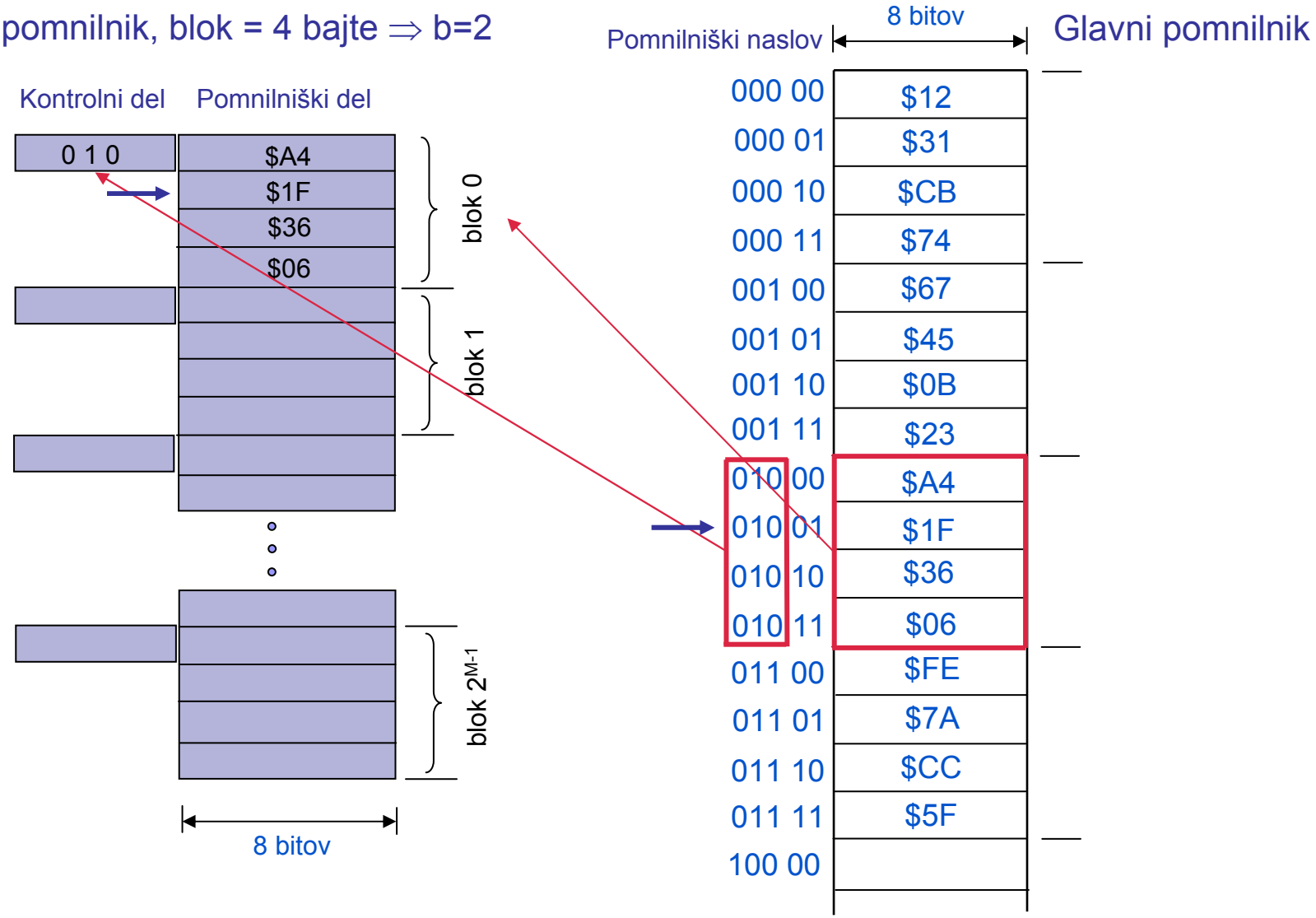
- Blok je nekaj sosednjih pomnilniških besed. Velikost bloka ($B = 2^b$) je tipično 4 do 512 pomnilniških besed.
- Kontrolni del vsebuje informacijo, ki enolično določa, kateri blok iz glavnega pomnilnika je v pomnilniškem delu, to je naslov bloka v glavnem pomnilniku.
- Vsak blok ima v predpomnilniku svoj kontrolni del, ki je skupen za cel blok.
- Med glavnim pomnilnikom in predpomnilnikom se vedno prenaša cel blok.

Predpomnilnik, blok = 4 bajte ⇒ b=2

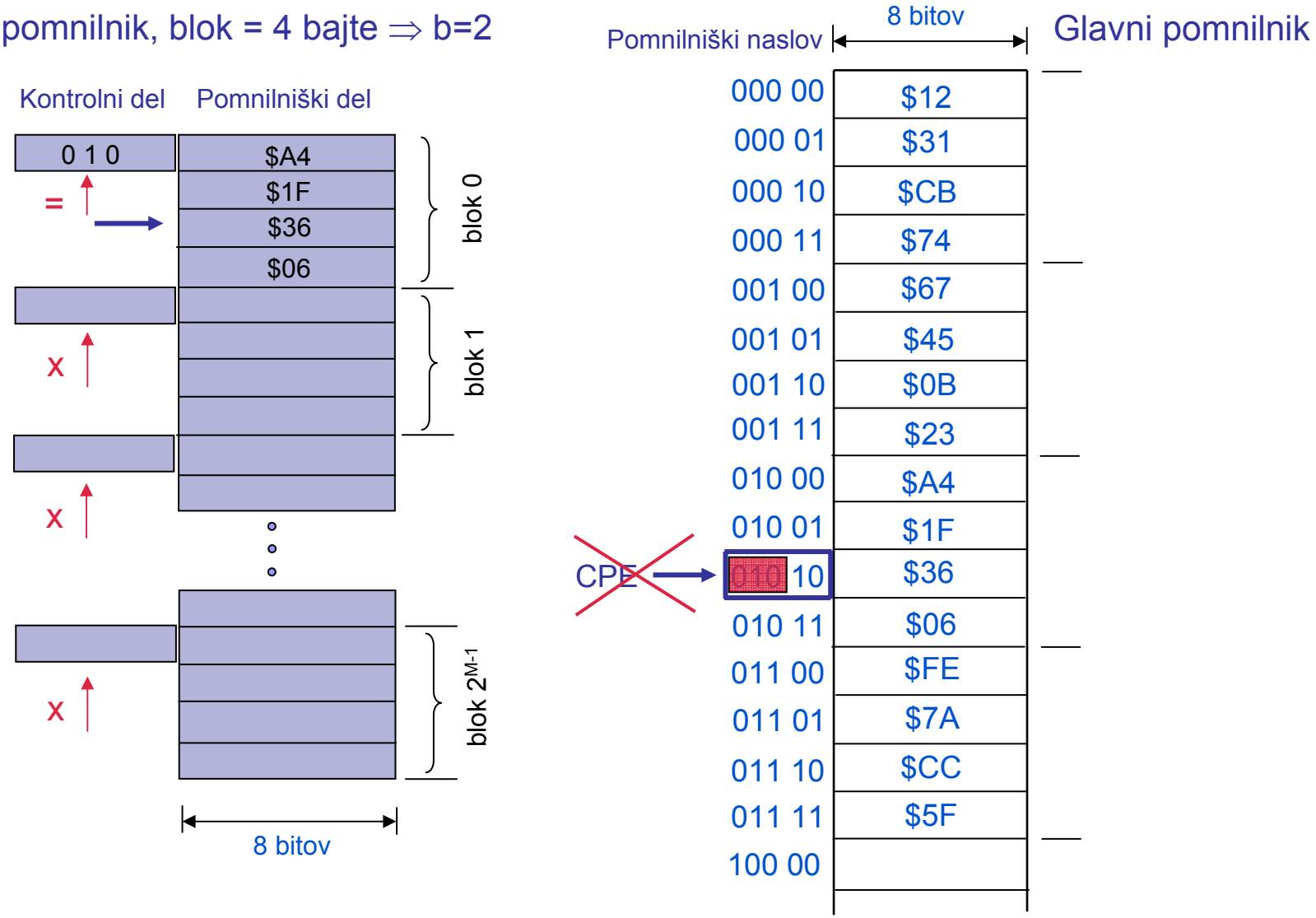




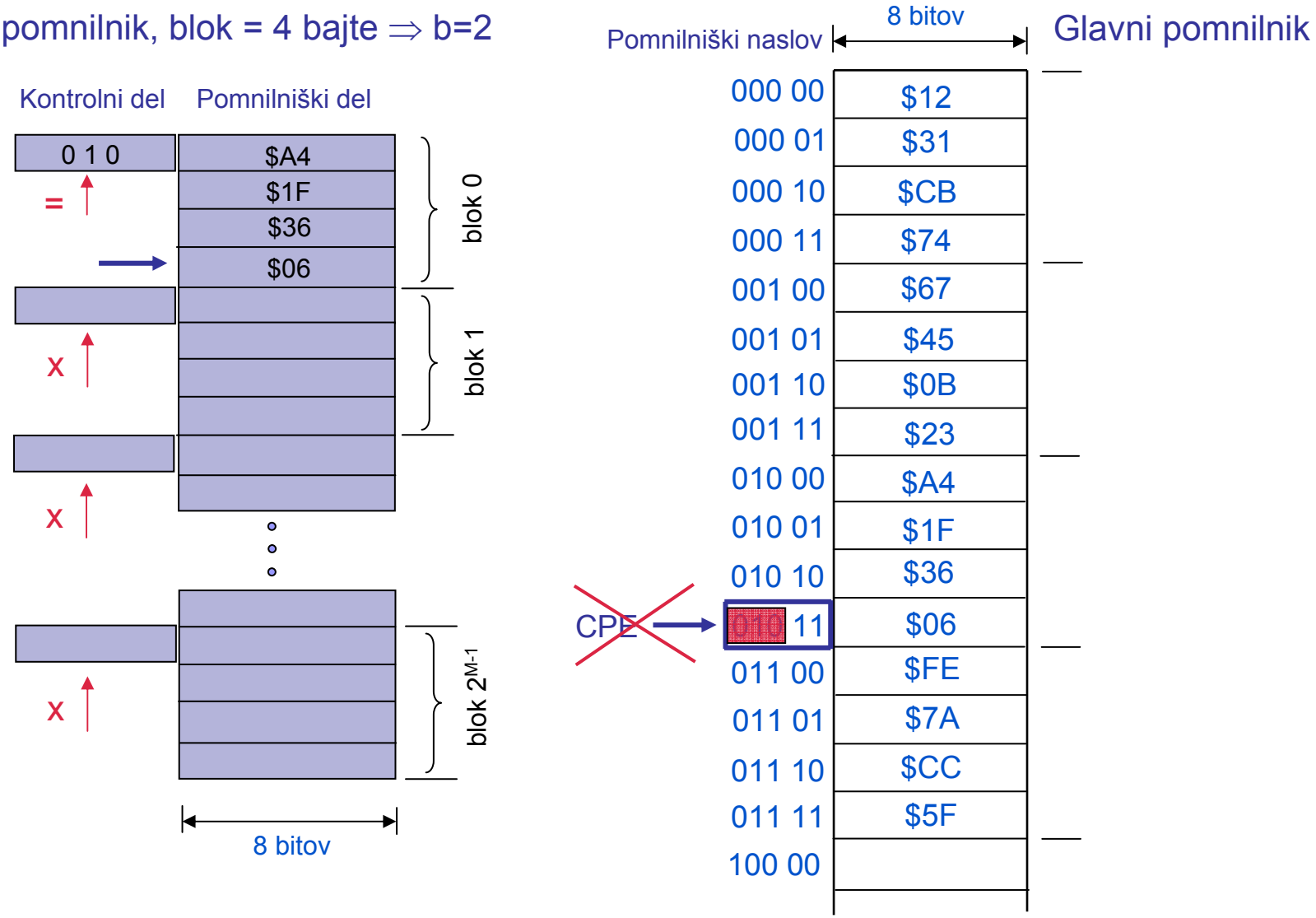
Predpomnilnik, blok = 4 bajte ⇒ b=2



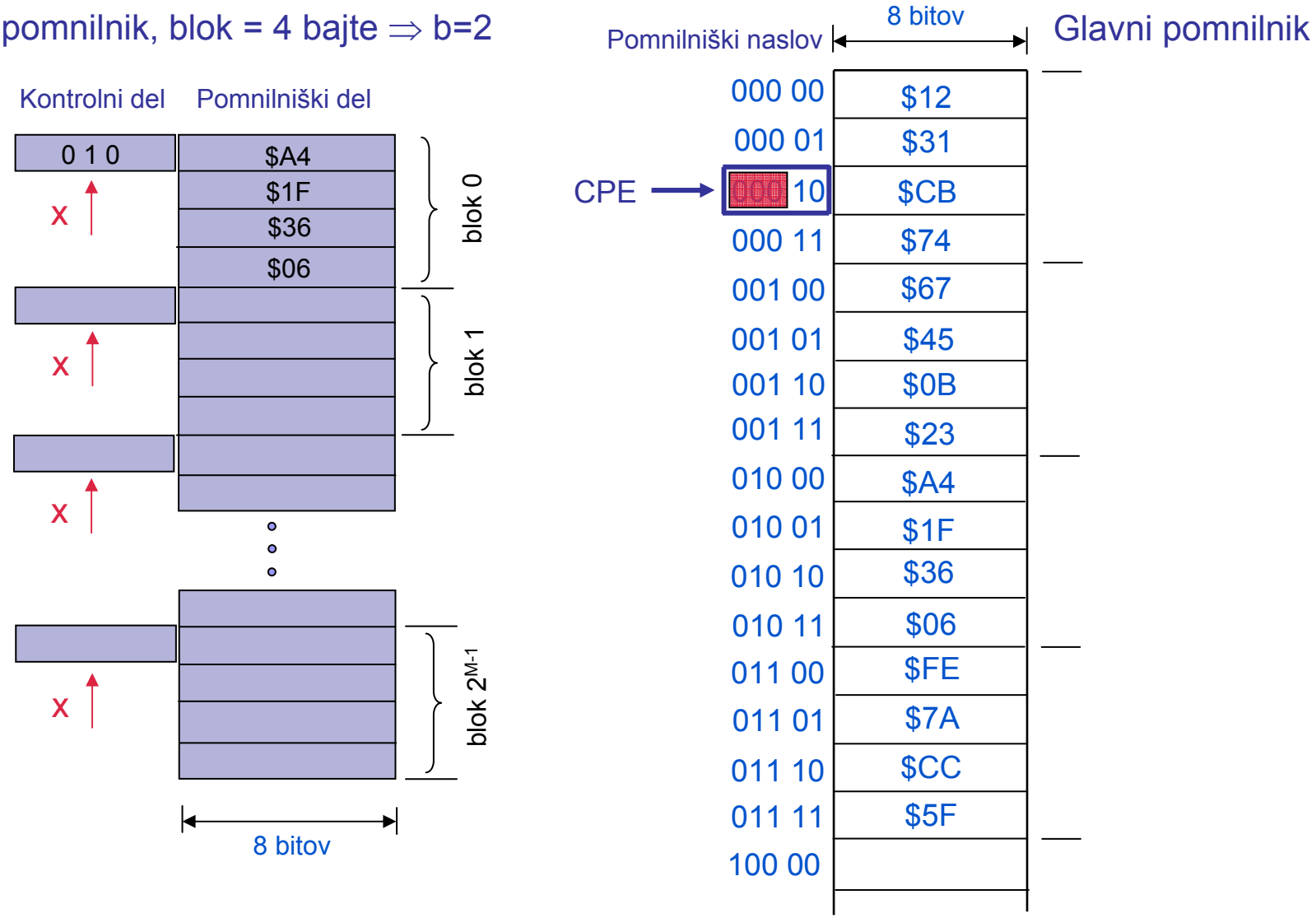
Predpomnilnik, blok = 4 bajte ⇒ b=2



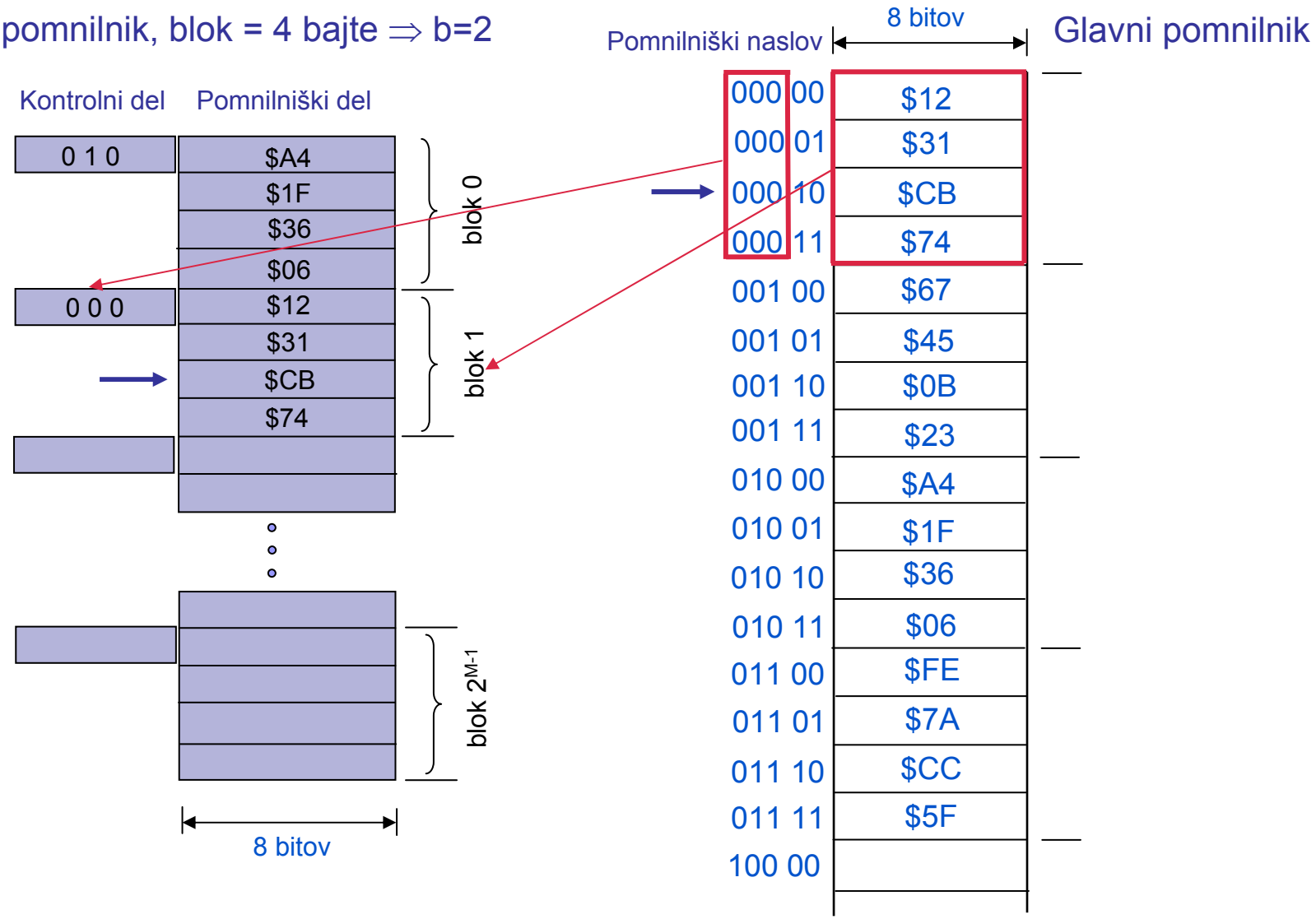
Predpomnilnik, blok = 4 bajte ⇒ b=2



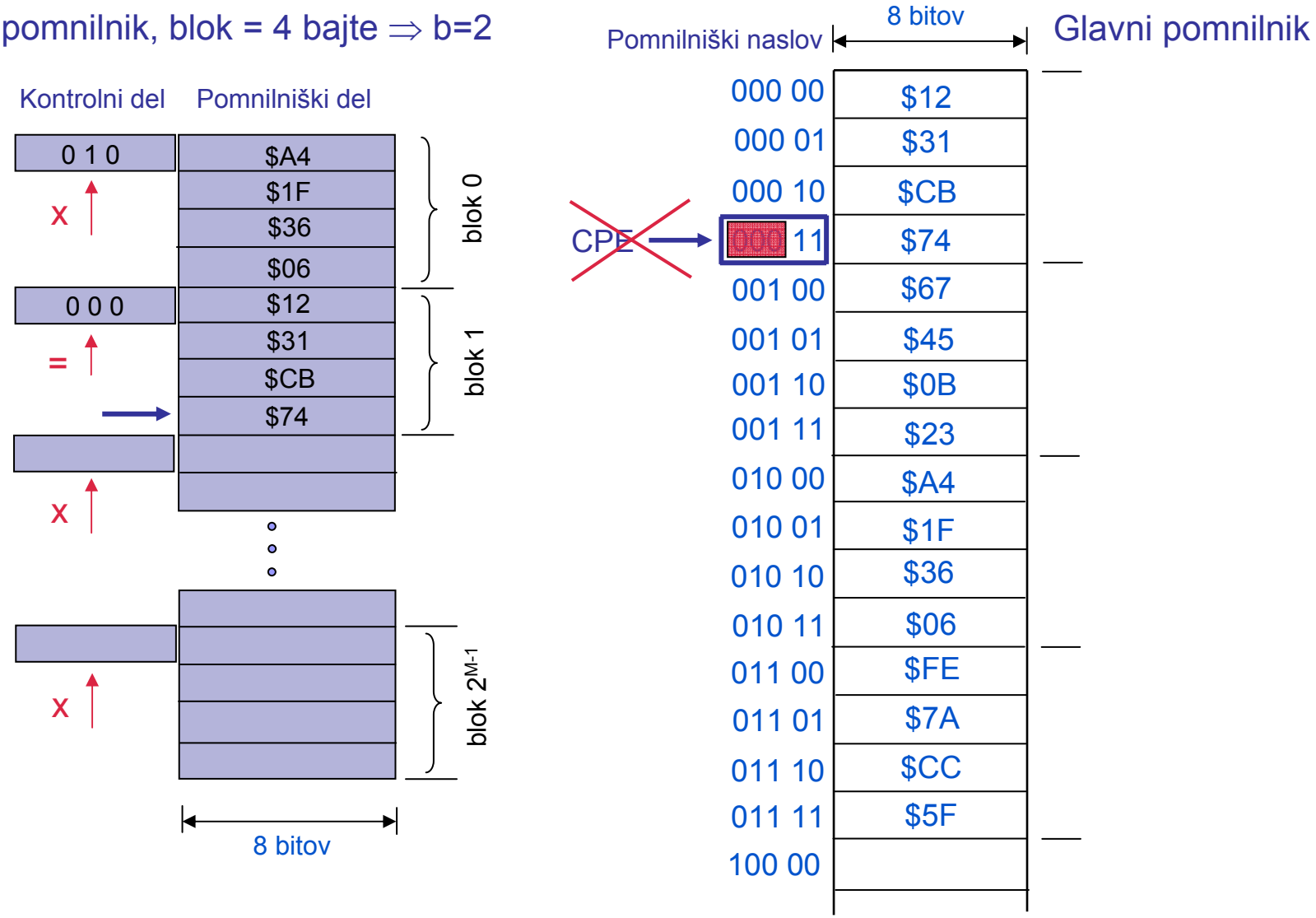
Predpomnilnik, blok = 4 bajte ⇒ b=2



Predpomnilnik, blok = 4 bajte ⇒ b=2



Predpomnilnik, blok = 4 bajte ⇒ b=2



- Pri vsakem dostopu CPE pošlje v pomnilnik *n*-bitni pomnilniški naslov. Zgornjih *n-b* bitov se primerja z naslovi v kontrolnih delih vseh blokov predpomnilnika (spodnjih *b*-bitov določa naslov besede znotraj bloka).
  - Če pri nekem bloku obstaja enakost, je iskana beseda v predpomnilniku - govorimo o zadetku v predpomnilniku in CPE izvrši dostop do besede v pomnilniškem delu predpomnilnika
  - Če enakosti ni, je to zgrešitev in iz glavnega pomnilnika se v predpomnilnik prenese blok v katerem je iskana beseda - **preslikava bloka**
  - Če so predpomnilniku vsi bloki zasedeni, novi blok zamenja enega od obstoječih - govorimo o **zamenjavi bloka**

- Potrebna je hitra primerjava vsebine kontrolnega dela predpomnilnika in dela pomnilniškega naslova, do katerega želi CPE dodstop.
- Če primerjava ni hitra, je dostop počasen, zato je treba pri preslikavi bloka iz glavnega pomnilnika v predpomnilnik vpelajti omejitve.
- Glede na strogost omejitev pri preslikavi razlikujemo tri vrste predpomnilnikov:
  - Čisti asociativni predpomnilnik
  - Set-asociativni predpomnilnik
  - Direktni predpomnilnik

- Razlika med temi tremi vrstami predpomnilnikov je predvsem v zgradbi kontrolnega dela.
- Pri čistem asociativnem in set-asociativnem predpomnilniku je kontrolni del zgrajen z asociativnim pomnilnikom, pri direktnem pomnilniku pa je kontrolni del navadni z naslovom naslovljiv pomnilnik.
- Pri čistem asociativnem predpomnilniku zato pri preslikavi bloka iz glavnega pomnilnika v predpomnilnik ni nobenih omejitev.

- To pomeni, da se poljuben blok iz glavnega pomnilnika lahko preslika v poljuben blok predpomnilnika (ker je iskanje po vsebini v kontrolnem delu hitro).
- Najstrožje omejitve pri preslikavi ima direktni predpomnilnik, kjer se določen blok iz glavnega pomnilnika lahko preslika samo v točno določen blok predpomnilnika (ker iskanje po vsebini v kontrolnem delu ni mogoče).
- Set-asociativni predpomnilnik je kompromis med čistim asociativnim in direktnim predpomnilnikom, zato so omejitve pri preslikavi delne.

- Določen blok iz glavnega pomnilnika se pri set-asociativnem predpomnilniku lahko preslika samo v določen set, znotraj seta pa omejitev pri preslikavi ni.

- Pomnilniški del je pri vseh treh vrstah predpomnilnikov navadni z naslovom naslovljiv pomnilnik, seveda čim hitrejši (SRAM).

Čisti asociativni predpomnilnik

- Kontrolni del predpomnilnika je narejen kot asociativni pomnilnik do katerega je dostop preko vsebine.
- Vsebina kontrolnega dela so naslovi blokov in predpomnilnik lahko na osnovi naslova, ki ga da CPE, takoj ugotovi ali je zadetek ali zgrešitev.
- Pri zadetku se opravi dostop do besede v bloku, ki je določena s spodnjimi *b*-biti pomnilniškega naslova.

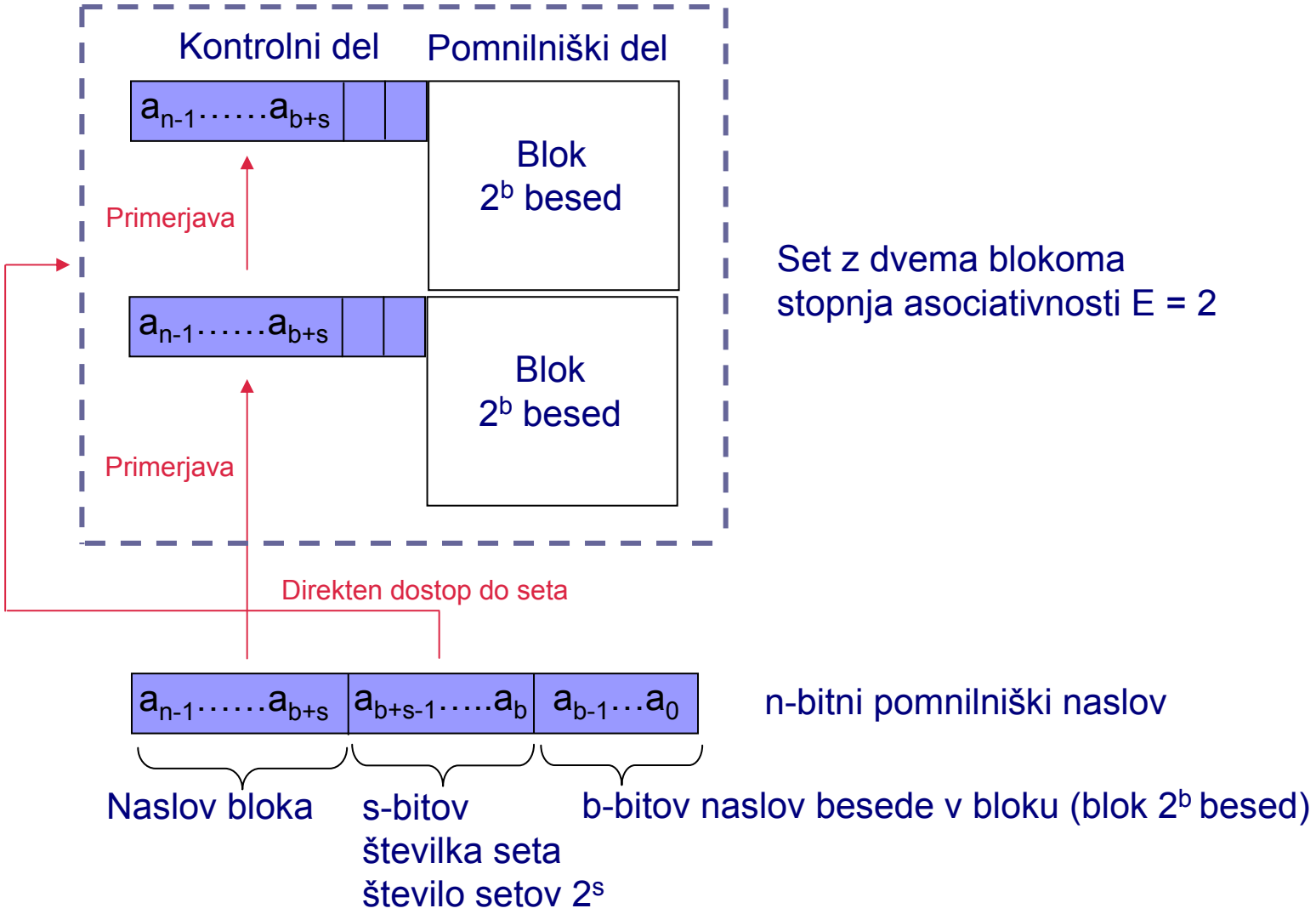
- Ker so vse besede asociativnega pomnilnika (kontrolnega dela) ekvivalentne, ni pri preslikovanju blokov iz glavnega pomnilnika v predpomnilnik nobenih omejitev.

- Asociativnih pomnilnikov z nekaj 10.000 besedami z današnjo tehnologijo ni mogoče narediti.
- Zato se čisti asociativni predpomnilniki danes uporabljajo samo v posebnih primerih, kjer so velikosti predpomnilnika samo nekaj 10 besed.
- Če želimo velik predpomnilnik, za kontrolni del namesto enega velikega asociativnega pomnilnika uporabimo več majhnih, ki so cenejši in jih lahko naredimo.

Set-asociativni predpomnilnik

- Če predpomnilnik razdelimo na **sete**, ki so vsak zase majhen asociativni predpomnilnik, dobimo set-asociativni predpomnilnik.
- Predpomnilnik razdelimo na $S=2^s$ setov, vsak set je majhen asociativni predpomnilnik.
- Število blokov v setu $E=2^e$ imenujemo **stopnja asociativnosti ali asociativnost**.
- Naslov seta v katerega se lahko preslika blok iz glavnega pomnilnika je: $S_i = Ai(b:n-1) \bmod 2^s$
  - *Ai(b:n-1) - zgornjih n-b bitov pomn. naslova = številka bloka*
  - 2^s *- število setov v predpomnilniku*

Predpomnilnik - direktni predpomnilnik



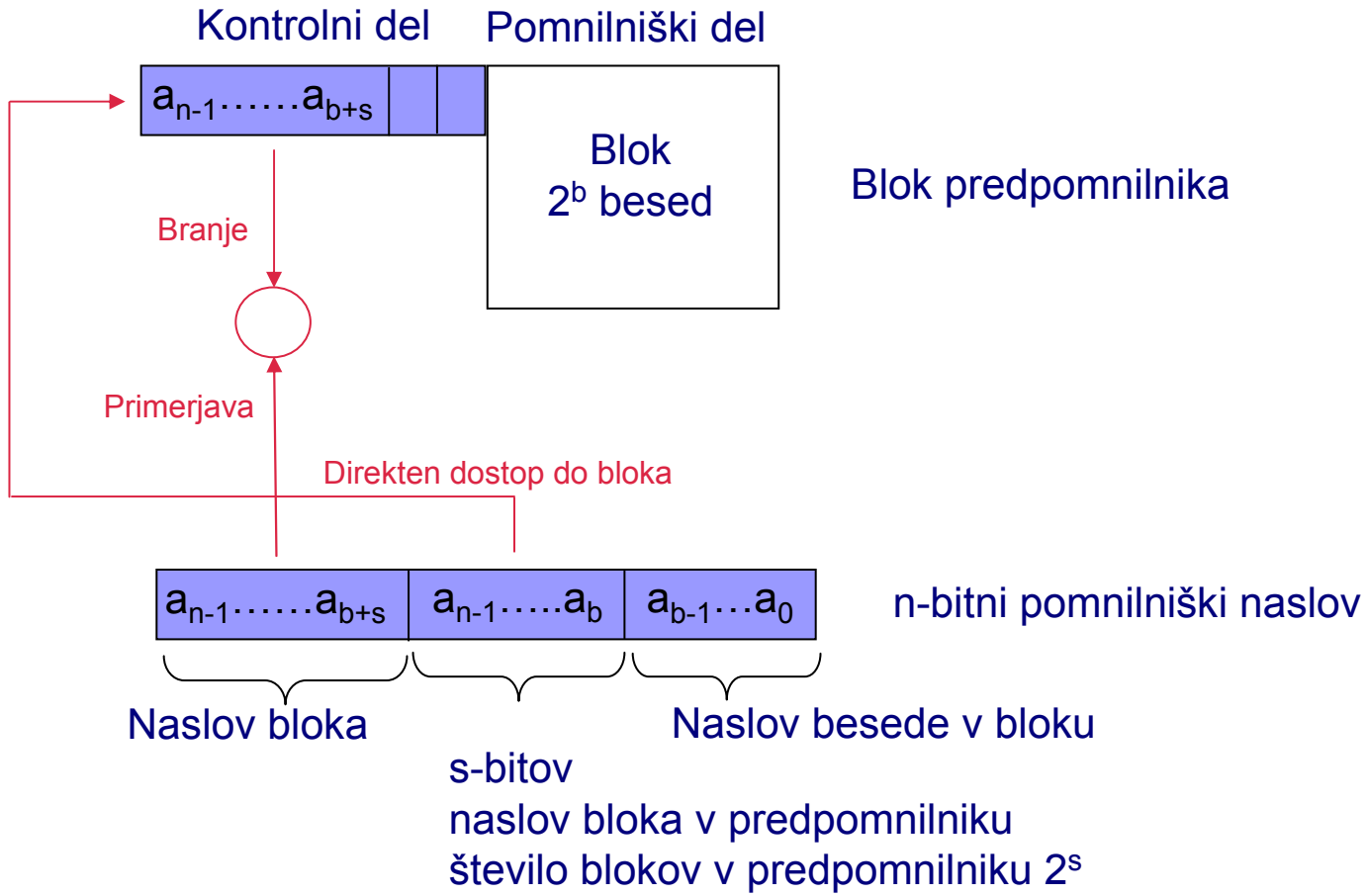
Set z dvema blokoma
stopnja asociativnosti E = 2

n-bitni pomnilniški naslov

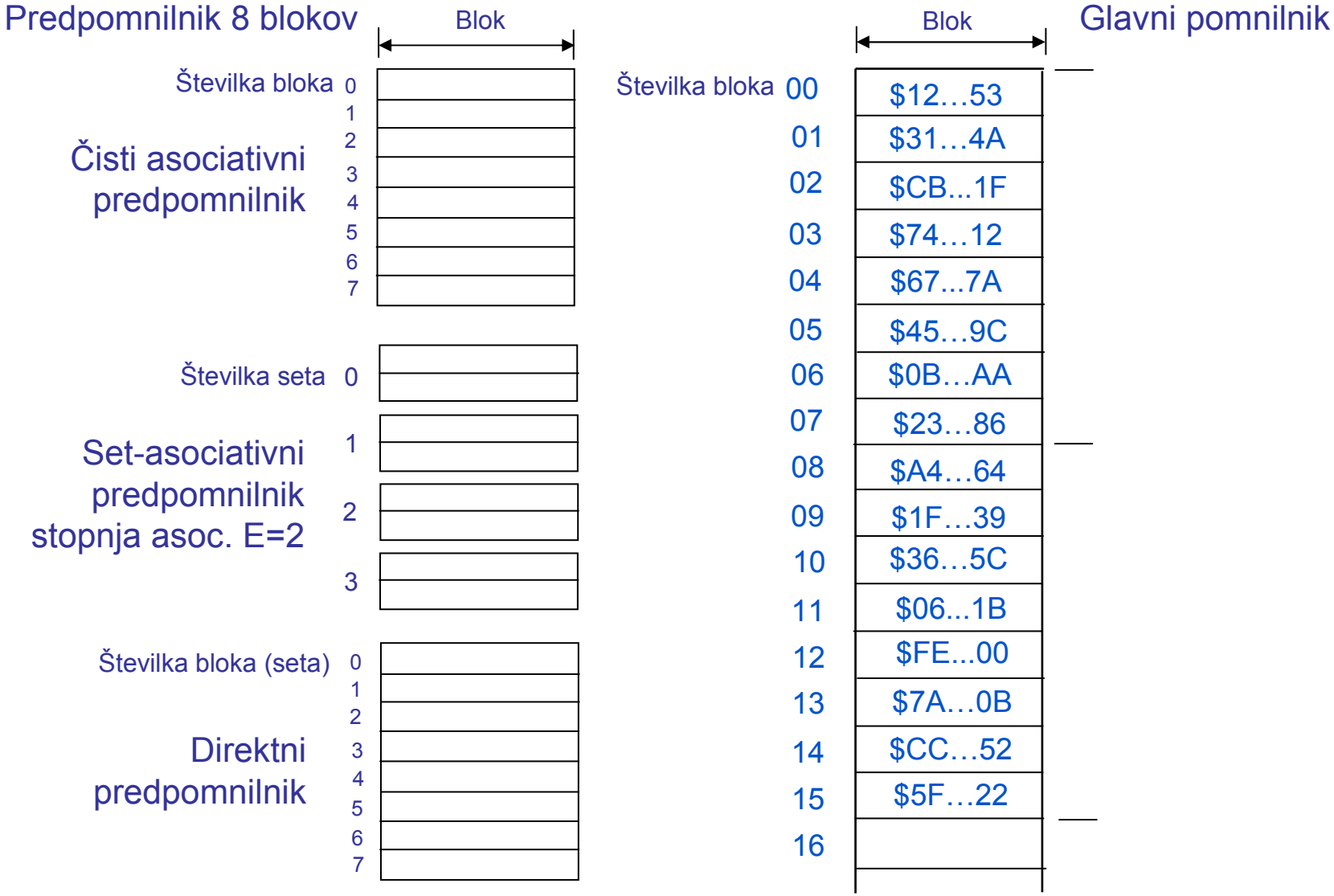
- Stopnja asociativnosti je velikost asociativnega pomnilnika s katerim je narejen kontrolni del vsakega seta.
- Stopnja asociativnost je tipično od 2 do 16 (2 do 16 blokov v setu $\Rightarrow$ 2 do 16 besed velik asociativni pomnilnik za kontrolni del seta)
- Velikost predpomnilnika je $M = S * E * B = 2^{s+e+b}$
  - $S = 2^s$ število setov
  - $E = 2^e$ stopnja asociativnosti (število blokov v setu)
  - $B = 2^b$ velikost bloka (število besed v bloku)

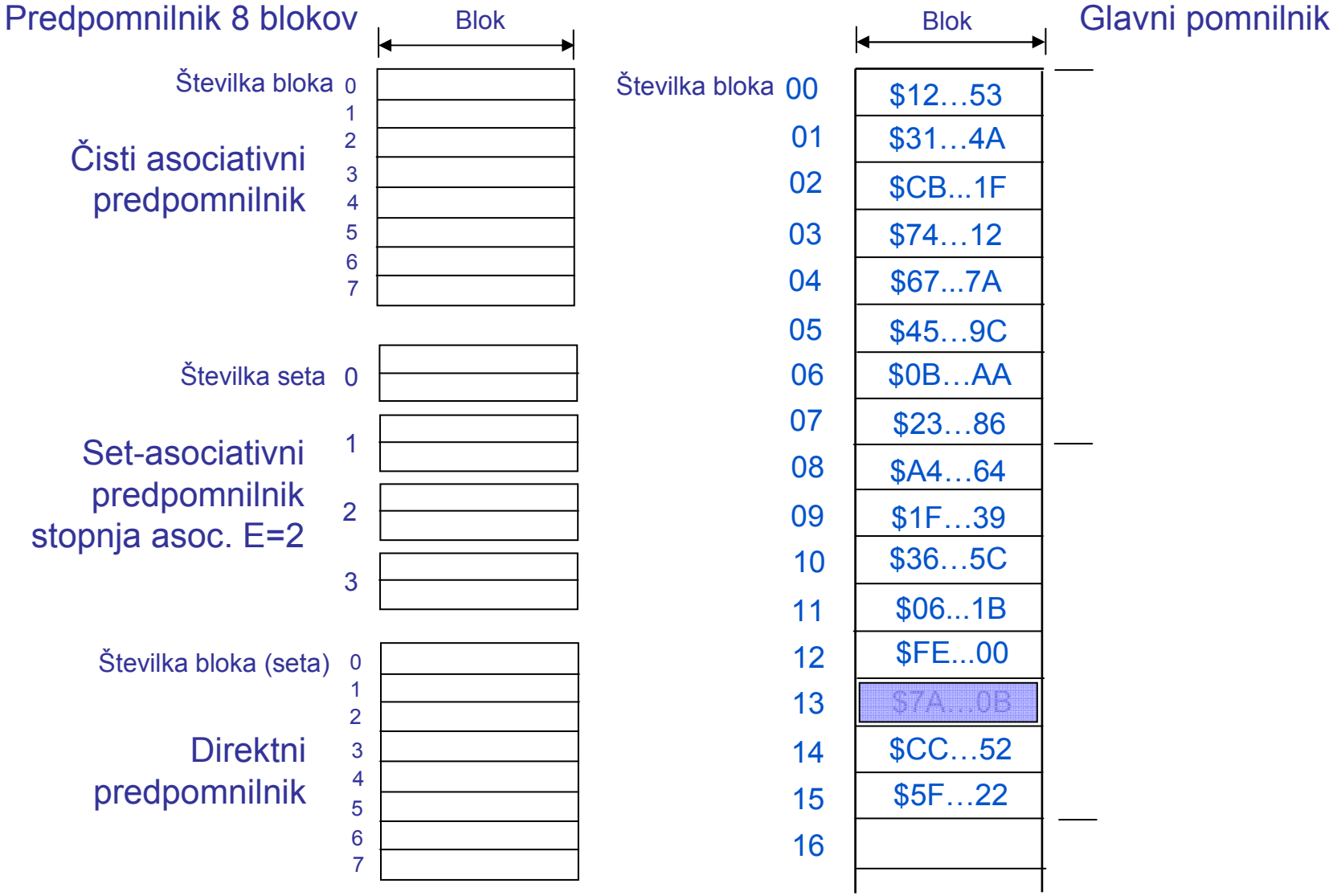
Direktni predpomnilnik

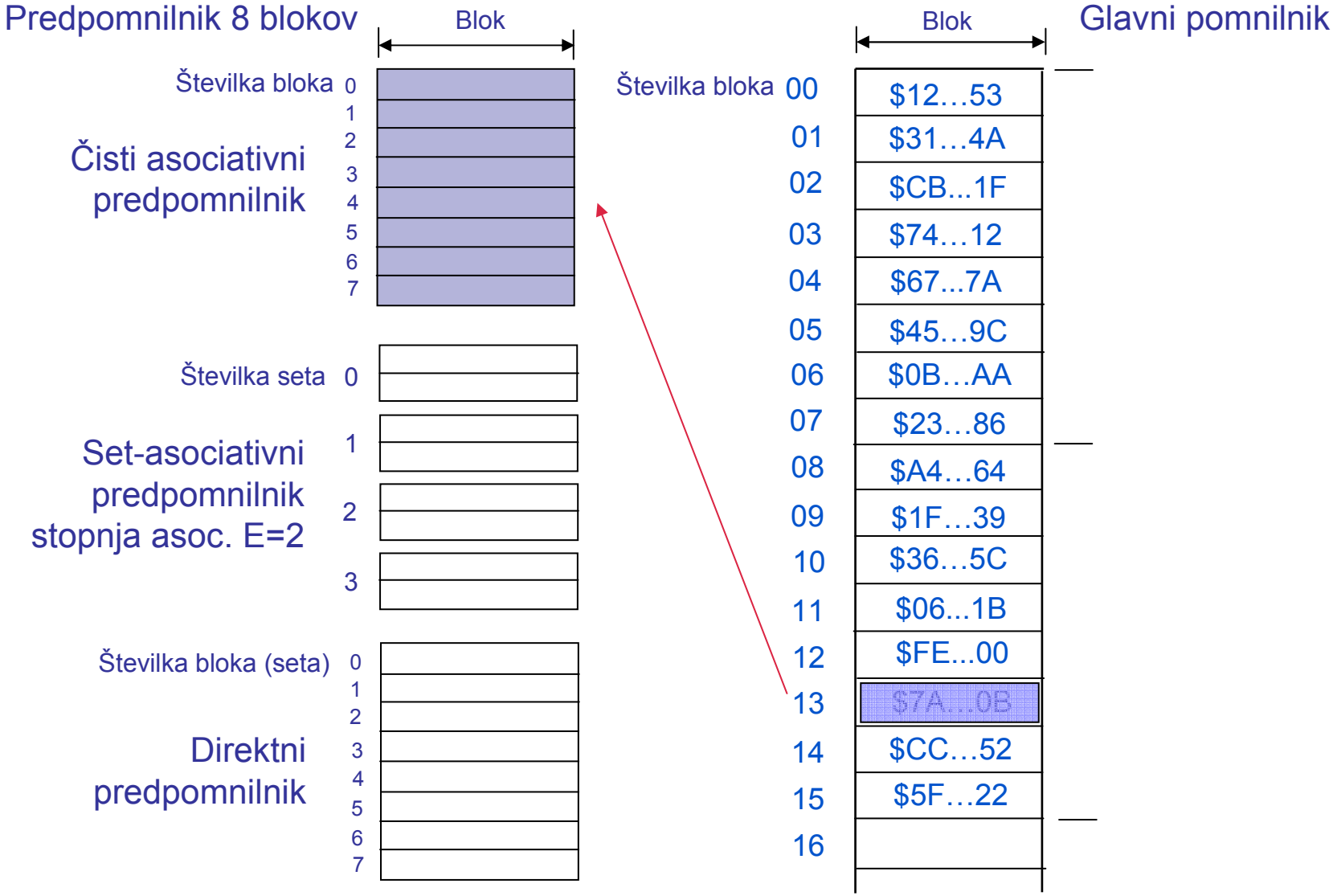
- Če je stopnja asociativnosti ena, je v vsakem setu samo en blok in za kontrolni del seta ne potrebujemo več asociativnega pomnilnika.
- Tak predpomnilnik je direktni predpomnilnik. Vsak blok iz glavnega pomnilnika se lahko preslika samo v točno določen blok predpomnilnika (znotraj “seta” ni več izbire, ker je v setu samo en blok)
- Naslov (številka) bloka v katerega se mora preslikati blok iz glavnega pomnilnika je: $S_i = Ai(b:n-1) \bmod 2^s$
  - *Ai(b:n-1) - zgornjih n-b bitov pomn. naslova = številka bloka*
  - *2^s - število blokov v predpomnilniku*

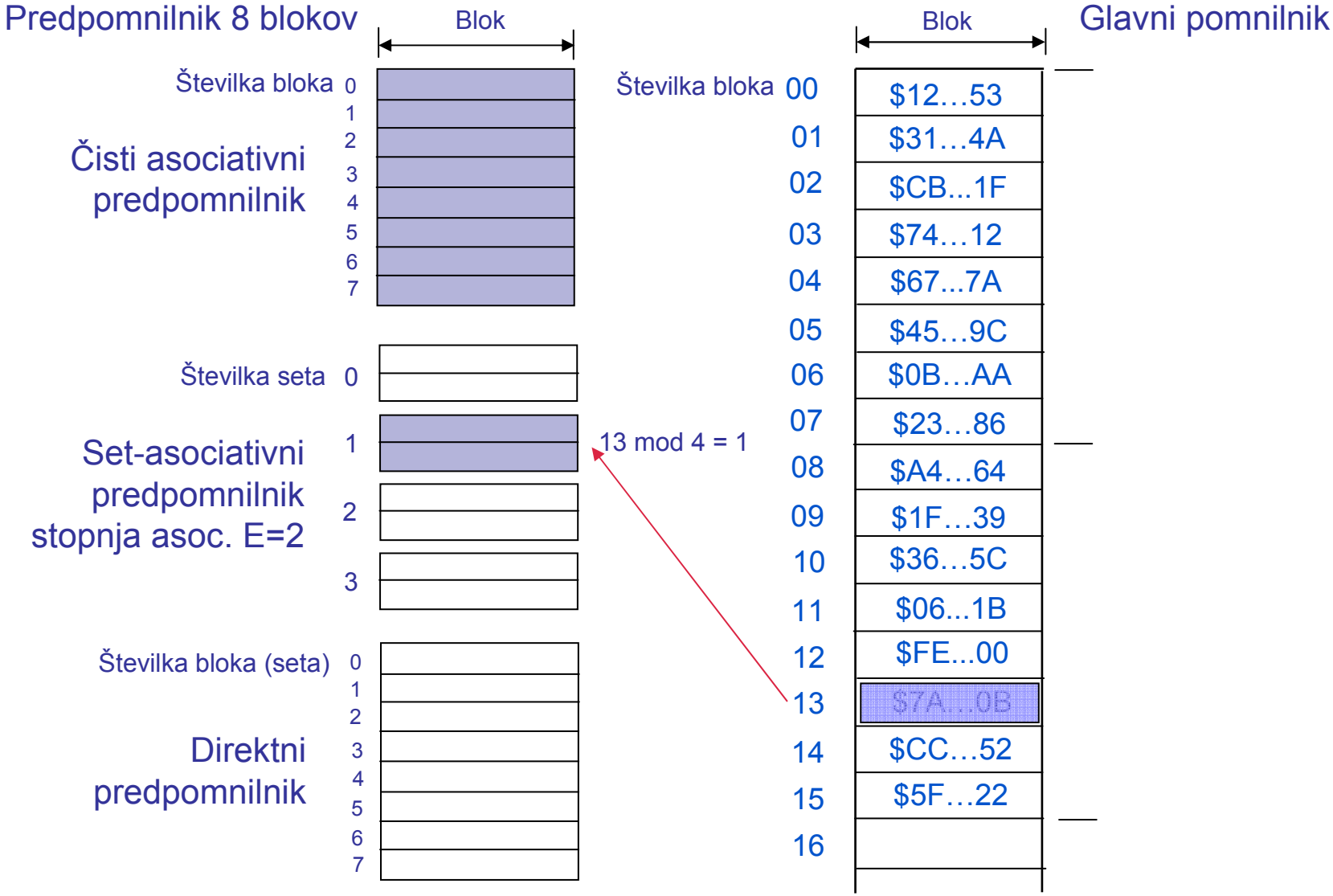


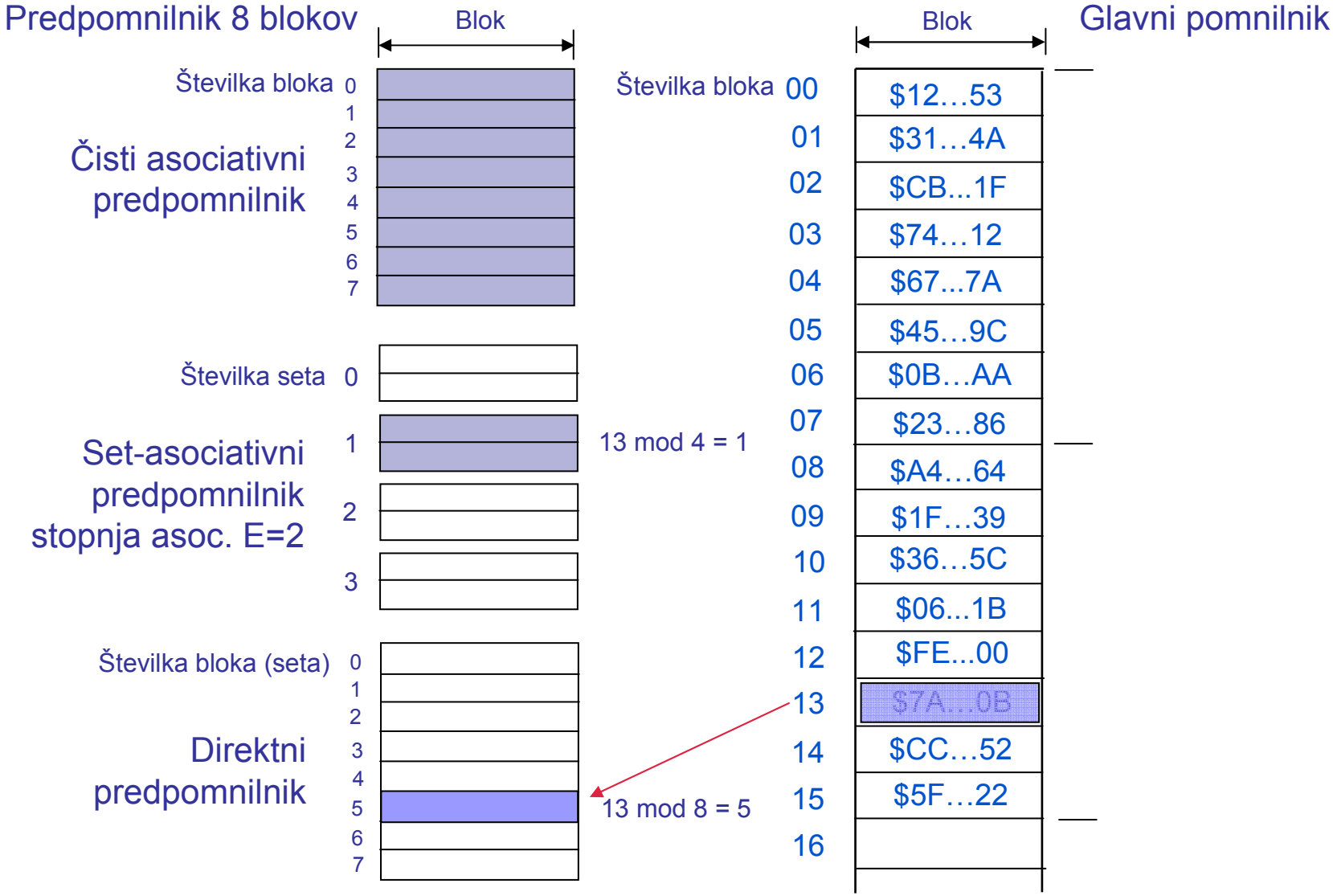
Predpomnilnik - preslikava bloka v predpomnilnik pri različnih vrstah predpomnilnikov











- Na verjetnost zadetka H vplivajo:
  - Velikost predpomnilnika M
  - Stopnja asociativnosti E
  - Velikost bloka B
- Predpomnilnik lahko povečamo tako, da povečamo:
  - Stopnjo asociativnosti E - v setu je več blokov, zato je večji kontrolni del (najdražje)
  - Število setov S
  - Velikost bloka B - kontrolni del ostane nespremenjen (najceneje)

- Večja kot je stopnja asociativnosti E, večja je verjetnost zadetka. Direktni predpomnilnik ima najmanjšo verjetnost zadetka. Ker pa je najcenejši, je lahko tudi velik, kar verjetnost zadetka popravi.

- Predpomnilniško pravilo 2 : 1
  - Verjetnost zadetka direktnega predpomnilnika velikosti M je približno enaka verjetnosti zadetka set-asociativnega predpomnilnika s stopnjo asociativnosti E=2 in velikostjo M/2.

- Verjetnost zadetka H z večanjem bloka do neke meje narašča, nato pa pada.
- Pri enaki velikosti predpomnilnika:
  - je s povečanjem velikosti bloka v bloku več sosednjih ukazov ali operandov (prostorska lokalnost)
  - hkrati se število blokov v predpomnilniku zmanjša (časovna lokalnost)
- Na verjetnost zadetka vpliva tudi izbira bloka, ki naj se zamenja pri zgrešitvi

- Pri direktnem predpomnilniku izbire ni, zamenja se blok, ki je določen s preslikavo.
- Pri čistem asociativnem in set-asociativnem predpomnilniku se blok lahko preslika v poljuben blok seta. Ko je predpomnilnik poln, potrebujemo strategijo za določanje bloka, ki naj se zamenja.
- Uporabljajo se naslednje strategije:
  - **Naključna strategija** (Random) - blok se izbere naključno, verjetnost, da bo blok zamenjan je enaka za vse bloke

- **LRU strategija** (Least Recently Used) - zamenja se blok, ki najbolj dolgo ni bil uporabljen. Ta strategija je boljša od naključne, ker zmanjša nevarnost, da bi se zamenjal blok, ki je bil pred kratkim uporabljen
- **FIFO strategija** (First In First Out) - zamenja se blok, ki je najbolj dolgo v predpomnilniku

- Realizacija LRU strategije je precej zapletena, zato se običajno uporablja pri manjših stopnjah asociativnosti (E=2). Pri predpomnilnikih z višjo stopnjo asociativnosti se uporablja naključna ali FIFO strategija.

Branje in pisanje v predpomnilnik

- Pri delovanju računalnika lahko opazimo, da je bralnih dostopov veliko več kot pisalnih (4 : 1)
- Pri branju je hitra rešitev dokaj enostavna:
  - Branje bloka je lahko istočasno s primerjavo kontrolne informacije
  - Pri zadetku je podatek na voljo CPE takoj
  - Ob zgrešitvi je podatek neuporaben, kar pa ne povzroči škode
- Pisanje v predpomnilnik se lahko začne le, če se ugotovi zadetek. Pisanje spremeni vsebino bloka v predpomnilniku, kar pomeni da se razlikuje od vsebine v glavnem pomnilniku.

- Dve vrsti pisalnih strategij se med seboj razlikujeta po tem, kdaj se sprememba vsebine bloka v predpomnilniku odrazi tudi v glavnem pomnilniku:
  - **Pisanje skozi** (write through) - informacija se vedno piše v predpomnilnik in v glavni pomnilnik, tako je vsebina bloka v predpomnilniku in glavnem pomnilniku vedno enaka
  - **Pisanje nazaj** (write back) - informacija se piše samo v predpomnilnik. Vsebina bloka v predpomnilniku je lahko različna od vsebine v glavnem pomnilniku, pri zamenjavi je zato spremenjeni blok treba prenesti nazaj v glavni pomnilnik.

- **Umazani bit** (dirty bit) poseben bit v kontrolnem delu bloka, ki pove ali je bila vsebina bloka spremenjena. Ob prenosu bloka v predpomnilnik se postavi na 0, ko se v blok piše, se umazani bit postavi na 1.

- Pri zamenjavi bloka se morajo nazaj v glavni pomnilnik prenesti samo bloki, ki imajo umazani bit postavljen na 1.

Vrste zgrešitev v predpomnilniku

- Majhno število zgrešitev je ključnega pomena za dobro delovanje predpomnilnika.

- Za zgrešitev v predpomnilniku so trije osnovni vzroki in glede na to ločimo tri vrste zgrešitev:
  - Obvezne zgrešitve
  - Velikostne zgrešitve
  - Konfliktne zgrešitve

- Obvezne zgrešitve
  - Ob prvem dostopu do besede te ni v predpomnilniku. Tem zgrešitvam se ni mogoče izogniti, zato tudi zgrešitve prvega dostopa ali mrzle zgrešitve.
  - Velikost predpomnilnika, stopnja asociativnosti in zamenjalna strategija na te zgrešitve ne vpliva.
  - Pri večjih blokih je obveznih zgrešitev manj.

- Velikostne zgrešitve
  - Nastopijo zaradi končne velikosti predpomnilnika ko je predpomnilnik poln.
  - Z večanjem predpomnilnika se število teh zgrešitev manjša.

- Konfliktne zgrešitve
  - Nastopajo samo pri set-asociativnih in direktnih predpomnilnikih zaradi omejitev pri preslikavi
  - Do konfliktne zgrešitve pride, ko se zamenja blok, ki se bo kmalu spet uporabil
  - Do zamenjave pride, ker se morata oba bloka zaradi omejitev pri preslikavi preslikati v isti set (ali blok)
  - Če bi bil predpomnilnik čisti asociativni, konfliktnih zgrešitev ne bi bilo

Skladnost predpomnilnika

- Problem skladnosti predpomnilnika (cache coherency) nastopi, kadar se vsebina nekega bloka v predpomnilniku razlikuje od vsebine istega bloka v glavnem pomnilniku in v drugih predpomnilnikih.

- V računalniku s predpomnilnikom je treba zagotoviti, da zaradi neskladnosti ne pride do napak.

- Problem skladnosti nastopa predvsem pri pred-pomnilnikih, ki uporabljajo pisanje nazaj.

- Za napačno delovanje zaradi neskladnosti sta dva glavna vzroka:
  - Prenosi med V/I napravami in glavnim pomnilnikom. Če se prenos v ali iz glavnega pomnilnika nanaša na naslove, ki so tudi v predpomnilniku, lahko vsebini nista enaki
  - Pri računalnikih z več procesorji, ki imajo vsak svoj predpomnilnik, lahko obstaja več različnih kopij iste pomnilniške besede

- Rešitve:
  - V/I naprave so lahko priključene tako, da gredo prenosi skozi predpomnilnik - ta rešitev se malo uporablja, ker močno poslabša verjetnost zadetka ($\Rightarrow$ v predpomnilnik se vpisujejo V/I podatki, ki jih CPE redko potrebuje)

- Pred izvrševanjem V/I ukazov se s posebnimi ukazi, ki jih ima vsak procesor s predpomnilnikom:
  - pri V/I branju razveljavi vsebina predpomnilnika
  - pri V/I pisanju prenese v glavni pomnilnik vse umazane bloke

- Mehanizem (dodatna logika) za zagotavljanje skladnosti:
  - **Centralni imenik** (central directory) - informacija o blokih, ki so trenutno v predpomnilniku, je shranjena na enem mestu v centralnem imeniku, ki je vgrajen v krmilnik pomnilnika
  - **Vohunjenje** (snooping) - se uporablja kadar so procesorji in glavni pomnilnik priključeni na isto vodilo. Vsi procesorji znajo vohuniti (opazovati dogajanje na vodilu) za naslovi na vodilu

- Protokol MESI (Modified, Exclusive, Shared, Invalid) - vsak blok v predpomnilniku ima dva kontrolna bita, ki določata v katerem od teh štirih stanj je blok.
  - Protokol deluje na osnovi vohunjenja, ob zgrešitvi se novemu bloku v predpomnilniku dodeli stanje E ali S (odvisno ali je blok samo v tem predpomnilniku ali tudi v kakšnem drugem)
  - Protokol MESI je za programerja neviden in se uporablja v operandnem predpomnilniku in predpomnilniku L2. V ukazni predpomnilnik pisanje ni dovoljeno, zato poseben protokol ni potreben

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 (povprečno število urinih period za izvedbo enega ukaza)
- 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

- Čas izvajanja programa z *I* ukazi pa je:

$$CPE_{čas} = I\,(CPI_I + M_I(1 - H)Zgreš.kazen)\,t_{CPE}$$

- Zgrešitve v predpomnilniku imajo večji vpliv pri hitrih procesorjih z majhnim CPI, eden od načinov za zmanjšanje zgrešitvene kazni je drugi nivo predpomnilnika (predpomnilnik L2)

