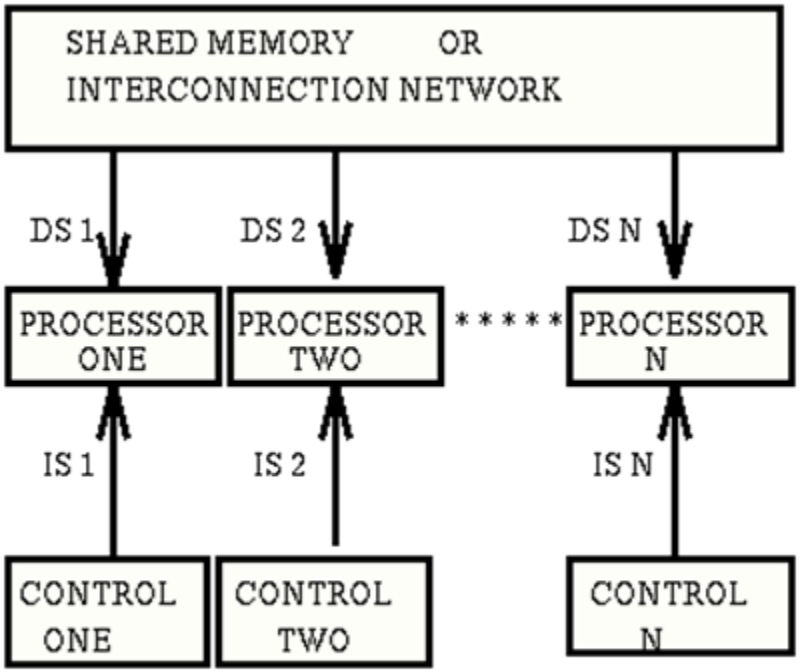
Paralelni računarski sistemi

MIMD

Karakteristike

- Najmoćnija i najopštija klasa paralelnih računara.
- MIMD sistem se sastoji od dva ili više procesora približno jednake moći izračunavanja.
- U MIMD sistemu ne postoji centralna upravljačka jedinica (CU – Control Unit.)
- Svaki procesor ima svoju CU, lokalnu memoriju i ALU.
- Radom svakog procesora upravljaju instrukcije koje izdaje sopstvena CU.
- Procesori mogu izvršavati različite delove programa nad različitim skupom podataka koji zajedno predstavljaju deo globalnog problema koji treba da se reši.
  - Drugim rečima, svaki procesor izvršava odredjeni skup zadataka (taskova ili procesa) koji predstavljaju deo nekog ukupnog posla (job)

Opšta struktura



DS = DATA STREAM IS = INSTRUCTION STREAM

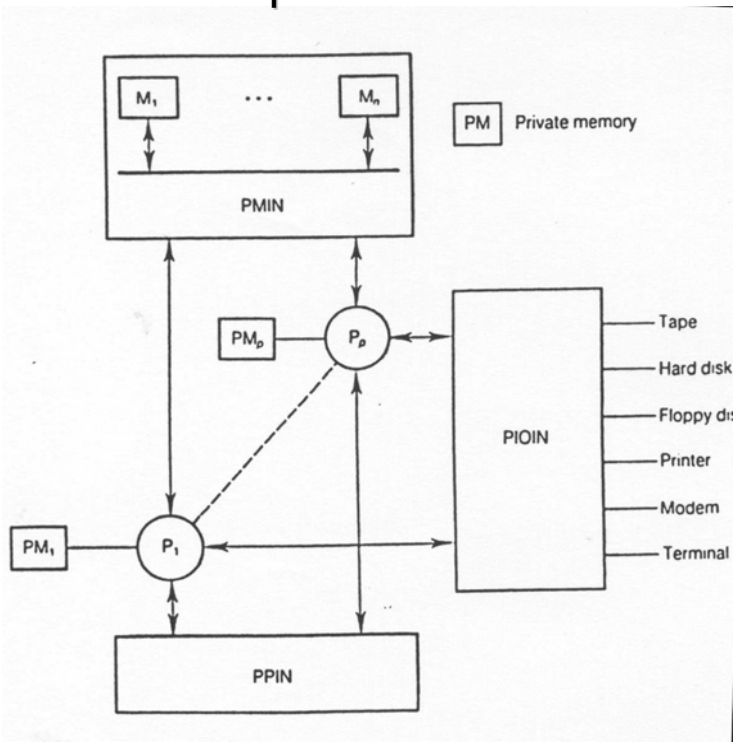
MIMD karakteristike

- Razmena podataka izmedju procesora (komunikacija) može da se obavi
  - preko zajedničke (deljive) memorije korišćenjem zajedničkih (deljivih) promenljivih,
  - ili eksplicitnim slanjem poruka izmedju procesora.
- Procesori u MIMD sistemu rade asinhrono, pa je potrebno obezbediti mehanizme za sinhronizaciju.
  - Sinhronizacijom se obezbedjuje korektan redosled izvršenja zadataka i uzajamno isključivo pravo pristupa deljivim resursima.
    - Mehanizmi iskorišćeni za komunikaciju mogu se iskoristiti i za sinhronizaciju, samo što se u ovom, drugom, slučaju umesto podataka razmenjuju upravljačke informacije.

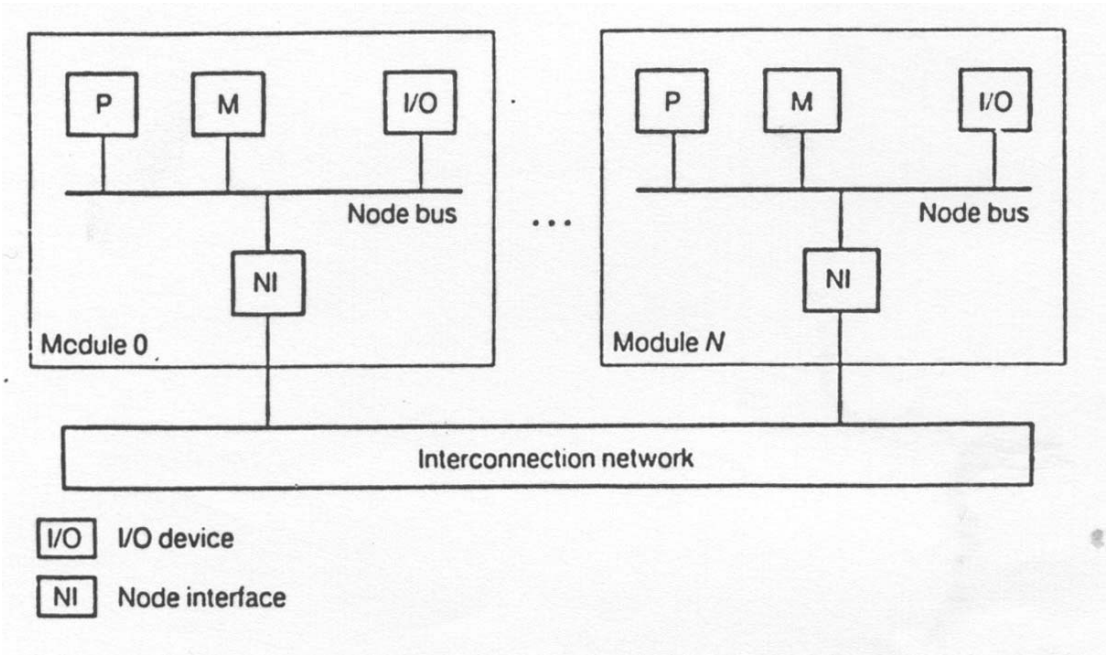
MIMD - podela

- MIMD sistemi se mogu podeliti na dve velike klase:
  - Čvrsto spregnuti sistemi (multiprocesori)
  - Slabo spregnuti sistemi (multiračunari

multiprocesor



multiračunar



Karakteristike multiprocesora

- Multiprocesor sadrži dva ili više homogenih procesora podjednakih mogućnosti;
- Svi procesori dele pristup zajedničkoj (deljivoj) memoriji, mada svaki procesor može imati malu lokalnu memoriju (memorije označene sa PM na Sl.1) koja se može koristiti za pamćenje jezgra OS.
- Sprega procesora sa zajedničkom memorijom ostvarena je preko procesor-memorija sprežne mreže (PMIN na sl.);
- Komunikacija izmedju procesora ostvaruje se preko deljive memorije korišćenjem zajedničkih (deljivih) promenljivih;
- Razmena upravljačkih informacija može se ostvariti i preko procesor-procesor sprežne mreže (PPIN), mada nije obavezno postojanje ove mreže;
- Svi procesori dele pristup U/I kanalima i uredjajima, DMA kontrolerima preko PIOIN mreže;

Karakteristike multiprocesora (nast.)

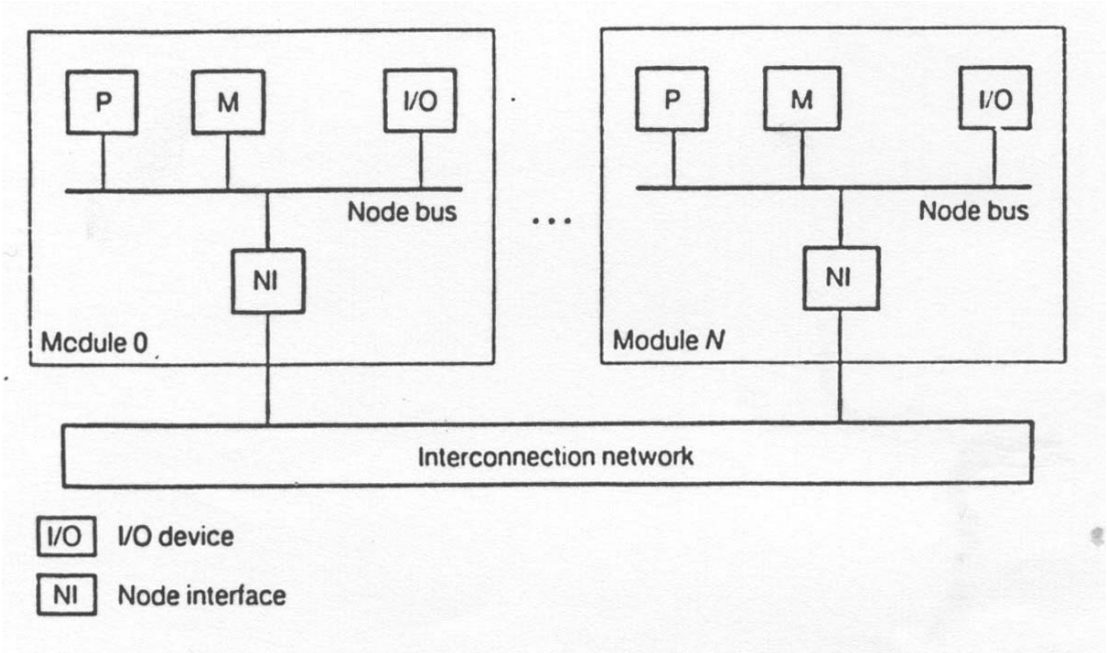
- Radom celog sistema upravlja jedinstveni OS koji obezbedjuje interakciju izmedju procesora i njihovih programa na nivou posla, zadataka, skupa podataka i elementa podataka.
- Multiprocesori mogu biti realizovani korišćenjem samo jedne sprežne mreže (PMIN).
  - Sprežne mreže multiprocesorskih sistema su dinamičkog tipa, što znači da se veze izmedju procesora i memorija uspostavljaju po zahtevu u toku izvršenja programa.
  - Najjednostavnija sprežna mreža je zajednička magistrala.
    - Cena i složenost ove sprežne mreže je najmanja (reda O(p), ako ima p procesora) ali je i propusnost najniža.
    - Na drugoj strani je crossbar sprežna mreža, koja ima potunu povezanost, ali i najveću cenu (reda $O(p^2)$, ako ima p procesora i p memorija).
    - Izmedju ova dva ekstremna rešenja postoje kompromisi koji se ogledaju u korišćenju više magistrala, jednostepenih i višestepenih sprežnih mreža.

Karakteristike multiprocesora (nast.)

- Može se desiti da dva ili više procesora upute zahtev za pristup istom memorijsom modulu što dovodi do konflikata.
  - Konflikte mora da razreši sprežna mreža.
    - Konflikti unose kašnjenje ilizbog sudra kod pristupa memorijskim modulima ili zbog vremena komutacije sprežne mreže.
    - Sa porastom sprežne mreže ovo kašnjenj može znatno umanjiti propusnost sistema.
- Broj obraćanja glavnoj memoriji i kašnjenje koje iz toga proizilazi, može se smanjiti uvodjenjem keš memorija izmedju sprežne mreže i procesora.
  - Uvodjenje privatnih keš memorija dovodi do problema nekonzistentnosti podataka jer sada u sistemu može postojati više kopija istog podatka.
    - Do neusaglašenosti dolazi kada podatak u jednom kešu bude modifikovan a da pri tome druge kopije tog podatka (u glavoj memoriji i drugim keševima) ne budu ažurirane.

Karakteristike multiračunara

- Svaki procesor ima svoju lokalnu memoriju i svoj sopstveni skup U/I uredjaja, mada mogu deliti neke periferije
- Procesor sa svojom memorijom i U/I uredjajima zove se računarski modul ili čvor.
  - Sprega procesora sa lokalnom memorijom i lok. U/I uredjajima ostvaruje se prko lokalne magistrale.
- Skup procesora može biti heterogen
- Kod slabo spregnutih sistema ne postoji zajednička memorija
- Komunikacija izmedju procesora se ostvaruje slanjem poruka kroz sprežnu mrežu.
  - Sprega čvora sa sprežnom mrežom ostvarena je preko odgovarajućeg interfeisa (NI na sl.)



Karakteristike multiračunara (nast.)

- Sprežne mreže su statičkog tipa.
- Moderni multiračunarski sistemi koriste hardverske rutere za slanje poruka.
  - Svaki čvor je priključen na jedan ruter.
    - Granični (krajnji) ruter može biti priključen na U/I.
    - Slanje poruka izmedju bilo koja dva čvora zahteva korišćenje niza rutera i kanala.
- Najpoznatije topologije sprežnih mreža kod multiračunara
  - prsten, stablo, mreža, torus, hiperkub.
  - Izmedju čvorova se mogu zahtevati različiti oblici komunikacija
    - Jedan-na-jedan, broadcast (emisija), multicast (selektivna emisija).

Multiprocesori – sprežne mreže

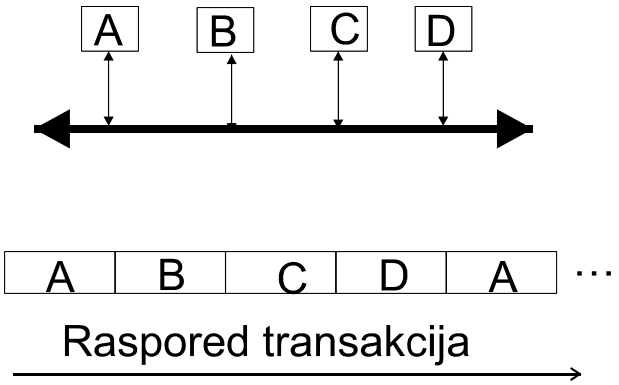
- Dinamičkog tipa
- Najjednostavnija sprežna mreža za povezivanje više procesora -vremenski deljiva (zajednička) magistrala.
  - Na ovaj jedinstveni sprežni put spregnute su sve komponente sistema: procesori, memorije i U/I uredjaji.
  - Ovakva sprežna mreža je potpuno pasvna jedinica bez aktivnih komponenti kao što su komutatori.
  - Operacijama prenosa po magistrali u potpunosti upravlja interfeis izvorne i odredišne jedinice.
    - Jedinica koja želi da inicira prenos (procesor ili U/I uredjaj) mora prvo da utvrdi status magistrale (raspoloživa ili ne), zatim
    - da adresira odredišnu jedinicu i utvrdi njenu raspoloživost i spremnost da prihvati podatke.
    - Nakon završetka prenosa magistrala mora da se oslobodi

Arbitraža na magistrali

- Ploče koje se povezuju na magistralu mogu biti tipa
  - Gospodar – može inicirati prenos po magistrali
  - Sluga – odaziva se gospodaru
  - Neki uredjaji mogu biti i gospodar i sluga, ali ne jednovremeno.
  - Pošto se na magistralu priključuje veći broj potencijalnih gospodara, a u jednom trenutku samo jedan može dobiti magistralu na korišćenje, potrebno je obezbediti arbitracione mehanizme koji će omogućiti da se dodela magistrale obavi bez konflikata.
- Arbitraža se može izvesti kao
  - Statička
  - Dinamička

Statička arbitraža

- Raspored transakcija na magistrali izmedju potencijalnih gospodra vrši se po unapred definisanom redosledu
  - Obično je redosled dodeljivanja kružni



  - Prednost :
    - jednostavnost (tj. Ugradnja jednostavnog hardvera).
  - Nedostatak
    - ne uzima u obzir realne potrebe za prenosom.
    - Magistrala se dodeljuje i onom gospodaru koji nema potrebe za prenosom.

Dinamička arbitraža

- gospodar magistrale odredjuje dinamičkim putem, po zahtevu.
- Dodela i oslobadjanje magistrale ostvaruju se odredjenom politikom
- Politike dodele magistrale mogu biti zasnovane na
  - Prioritetu – svakom potencijalnom gospodaru dodeljen je fiksni prioritet.
    - U slučaju više zahteva magistrala se dodeljuje gospodaru sa najvišim prioritetom
  - Nepristrasnosti - potencijalni gospodari imaju jednak prioritet.
    - Svakom gospodaru koji je izdao zahtev mora se garantovati dodela magistrale pre nego što se bilo kom drugom gospodaru po drugi put dodeli magistrala.
    - prioriteti gospodra nisu fiksni, već se menjaju dinamički.
    - Onaj gospodar koji je poslednji koristio magistralu dobija najniži prioritet.
  - Kombinovana - politika zasnovana na prioritetu i nepristrasnosti.
    - Obično se U/I zahtevima vrši dodela zasnovana na prioritetima, a procesorskim zahtevima na politici nepristrasnosti.
    - Multiprocesorski sistemi standardno koriste kombinovanu politiku

Dinamička arbitraža (nast.)

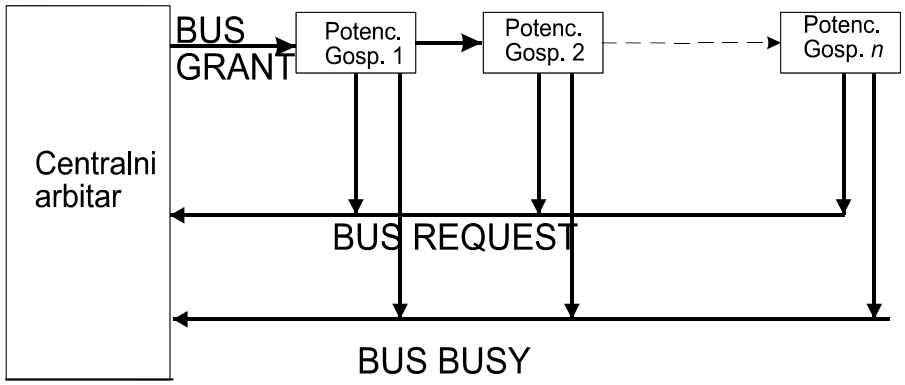
- Politike oslobadjanja magistrale
  - Oslobadjanje po zahtevu - tekući gospodar magistrale ima pristup, sve dok se ne generiše drugi zahtev
  - Oslobadjanje nakon obavljene transakcije – nakon obavljene transakcije gospodar oslobadja magistralu
  - Istiskivanje – gospodar koji ima viši prioritet u odnosu na tekućeg gospodra, uslovljava da gospodar sa nižim prioritetom oslobodi magistralu i ako nije završio sa prenosom
- Hardver za arbitražu može biti realizovan
  - centralizovan
  - distribuiran

Centralizovana arbitraža

- Kod centralizovane arbitraže hardver je koncentrisan na jednom mestu:
  - može biti u jednom od modula koji se povezuju na magistralu
  - poseban hardver koji se zove *arbitar magistrale*.
- Gospodar koji zahteva dodelu magistrale predaje zahtev centralnom arbitru.
  - Ako postoji veći broj zahteva, centralni arbitar na osnovu usvojene politike dodele odlučuje kome će dodeliti magistralu.
- Hardverski mehanizmi koji se koriste za dodelu i zahvatanje magistrale mogu se realizovati kao
  1. deljivi zahtevi i lančano zahvatanje
  2. nezavisni zahtevi i zahvatanja

deljivi zahtevi i lančano zahvatanje

- Svaki potencijalni gospodra magistrale izdaje zahtev za dodelu magistrale preko linije BUS REQUEST.
  - Svi zahtevi su povezani žičanom ILI logikom.



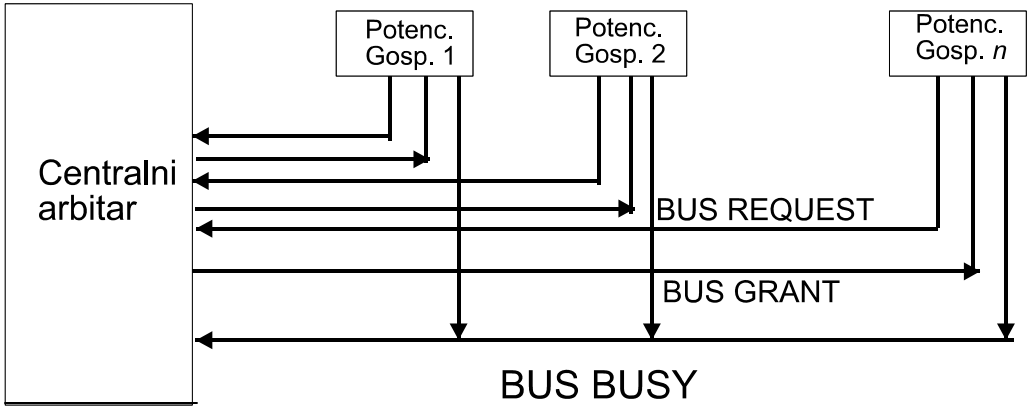
  - Kada centralni arbitar primi zahtev, on predaje signal BUS GRANT potencijalnom gospodaru magistrale označenom sa 1.
    - Linija BUS GRANT povezuje sve potencijalne gospodare u lanac (daisy chain), tako da gospodar 1 predaje signal gospodaru 2, itd.
    - Gospodar koji je izdao zahtev ne prenosi signal dalje (prekida lanac) i aktivira liniju BUS BUSY čime ukazuje da je zahvatio magistralu.
    - Kada tekući gospodar završi sa prenosom oslobadja liniju BUS BUSY i naredni ciklus arbitraže može da počne.

deljivi zahtevi i lančano zahvatanje (nast.)

- Očigledno je da je ovde politika dodele zasnovana na fiksnim prioritetima.
  - Gospodar koji je bliži centralnom arbitru ima viši prioritet.
- Prednost ove šeme je jednostavnost izvodjenja i mali broj linija zahtevanih za arbitržu.
- Nedostatak je što je priorite odredjen fizičkom pozicijom:
  - može se desiti da gospodar sa nižim prioritetom ne dobiju magistralu ako se često izdaju zahtevi od strane gospodara sa nižim prioritetom

Nezavisni zahtevi i zahvatanja

- Svakom potencijalnom gospodaru magistrale dodeljuje se posebna linija za izdavanje zahteva i posebna linija za zahvatanje preko kojih se povezuju sa centralnim arbitrom.



- Kada potencijalni gospodar želi dodelu magistrale, on postavlja svoju liniju BUS REQUESTi.
- Arbitar odabira potencijalnog gospodara i odabira odgovarajuću liniju BUS GRANTi.
- Odabrani gospodar briše svoj zahtev i aktivira liniju BUS BUSY.
  - U slučaju više zahteva, centralni arbitar vrši dodelu magistrale na osnovu usvojene politike dodele

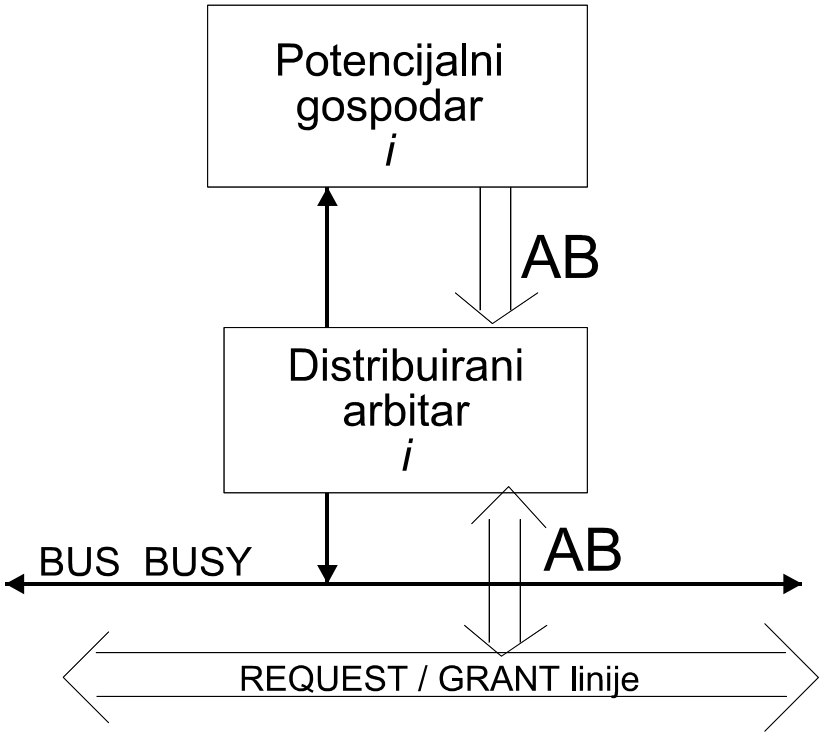
Nezavisni zahtevi i zahvatanja (nast.)

- Prednost
  - kratko vreme arbitraže,
- nedostatak
  - veliki broj linija za povezivanje potencijalnih gospodra i centralnog arbitra.

Distribuirana arbitraža

- Hardver za arbitražu je raspodeljen po potencijalnim gospodarima magistrale
  - Svaki potencijalni gospodar ima sopstveni arbitar i jedinstveni arbitracioni broj koji se koristi da razreši sudare kod jednovremenih zahteva za korišćenjem magistrale.
  - Kada dva ili više uredjaja konkurišu za korišćenje magistrale, pobednik je onaj koji ima viši arbitrcioni broj.
  - Svaki potencijalni gospodar može poslati svoj arbitracioni broj (AB) na deljive request/grant linije preko svog arbitra
  - Od svih zahteva se formira zbirni arbitracioni broj, #AB
  - Nakon ovoga, svaki arbitar poredi svoj AB sa zbirnim, počev od bita najveće težine.
    - Ako je njegov broj niži od zbirnog to znači da je njegov prioritet niži i zahtev se uklanja.
    - Na kraju na linijama ostaje arbitracioni broj onog gospodara koji ima najviši prioritet i njemu se dodeljuje magistrala

Distribuirana arbitraža

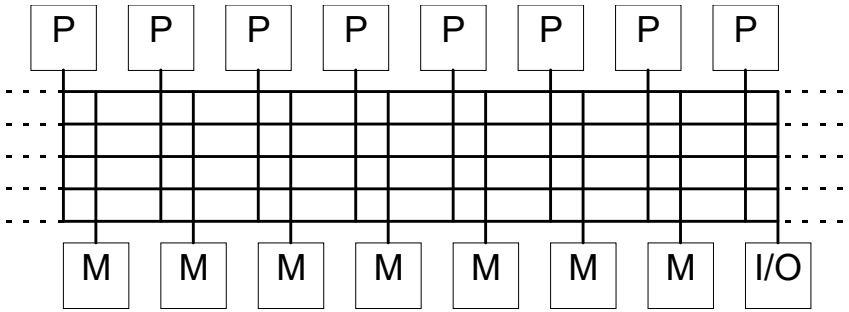


Zajednička magistrala - zaključak

- multiprocesorski sistemi koji koriste magistralu za povezivanje (npr. Sequent Symetry serija, Encore Multimax serija) imaju mali ili srednji broj procesora (do 32).
- Ekspanzija sistema dodavanjem procesora povećava saobraćaj namagistrali, što smanjuje propusnost sistema i povećava arbitracionu logiku.
- Ukupna brzina prenosa unutar sistema ograničena je širinom i brzinom magistrale.
- Zbog toga je prisustvo lokalnih keš memorija veoma poželjno

Više magistrala

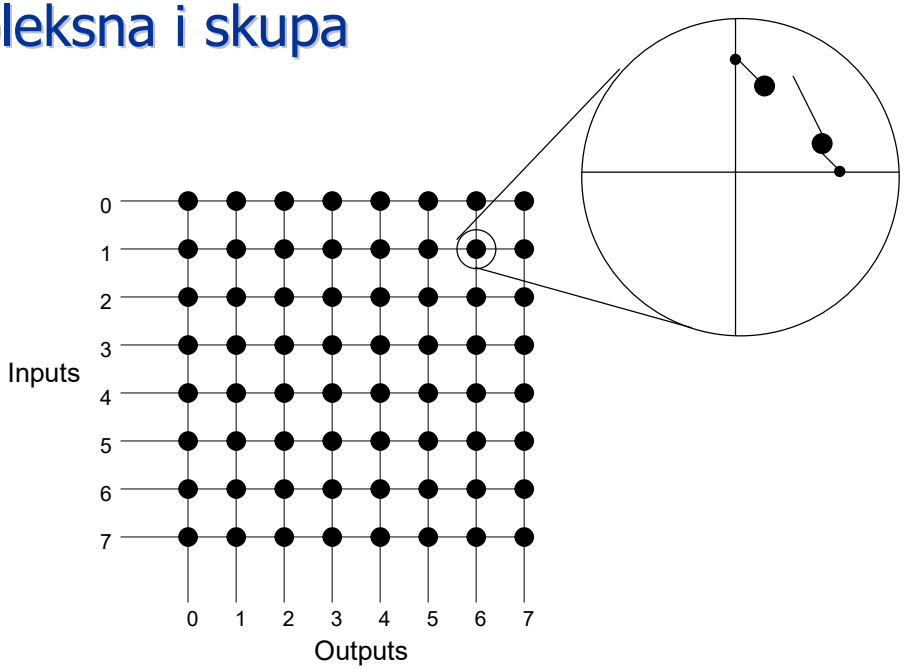
- Performanse sistema se mogu poboljšati korišćenjem više magistrala



- Prenos se može obavljati simultano po svakoj magistrali, tako da više procesora može pristupati jednovremeno različitim memorijskim bankama.
- Ako je potrebno obaviti više transakcija nego što ima magistrala, opet je neophodna arbitraža
- To povećava cenu sistema i potrebna je kompleksnija arbitražana logika.
- ako se broj magistrala poveća tako da za svaku mem. banku postoji posebna magistrala dobija se crossbar sprežna mreža

Crossbar mreža

- crossbar mreža je neblokirajuća, jednostepena SM
  - pruža mogućnost povezivanja svakog-sa-svakim.
  - emisija je moguća jer bilo koji ulaz može biti povezan sa svim izlazima
- Nedostatak
  - veliki broj komutacionih elemenata (ako je potrbno povezati n ulaza i m izlaza potrebno je $m \times n$ komutacionih elemenata
  - Kada je broj procesora i memeorijskih banaka veliki, mreža postavje veoma kompleksna i skupa

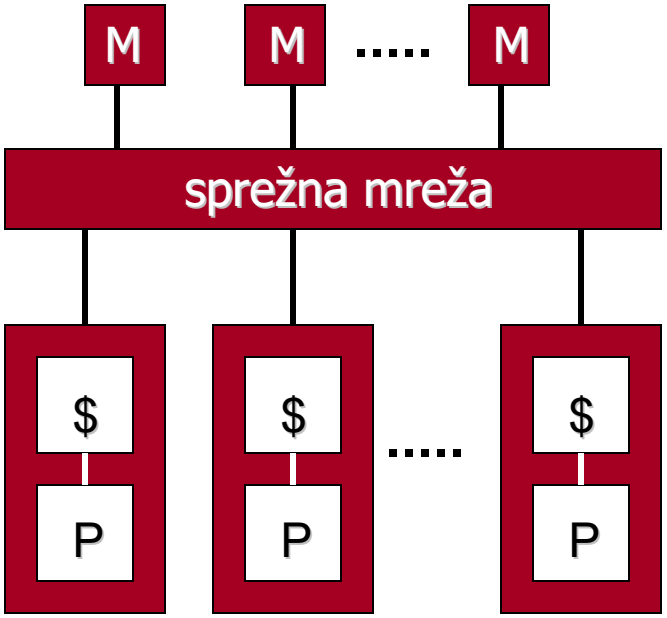


Keš koherencija

- Da bi se kod multiprocesorskih sistema otklonili problemi kao što su:
  - sudari kod pristupa zajednickoj memoriji
  - konflikti pri komunikaciji (ako više procesora zahteva isti sprežni put)
  - latentnost pristupa kroz sprežn umrežu
    - kod multiprocesorskih sistema sa velikim brojem procesora sprežna mreža je veoma kompleksna pa je latentnost kod takvih mreža velika)
- koriste se privatne keš memorije koje se pridružuju svakom procesoru

Keš koherencija

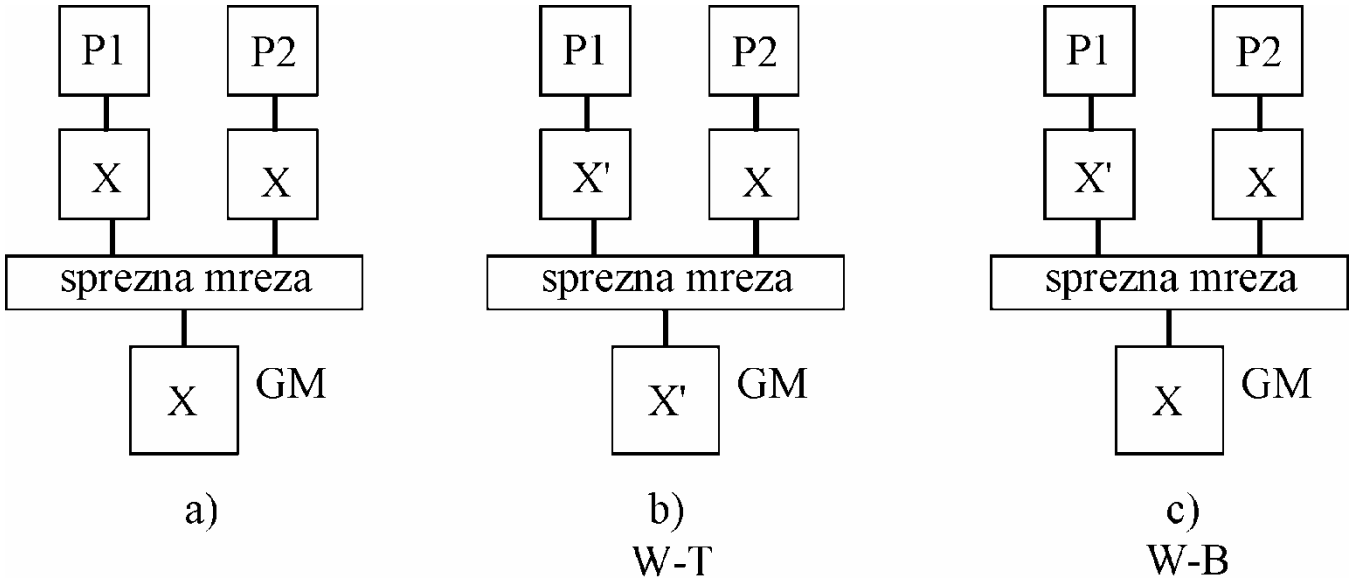
Struktura multi-procesorskog sistema sa keš memorijama



- Može se desiti da u jednom trenutku vise keš memorija poseduje kopiju istog podatka iz glavne memorije.
- Bilo koja lokalna modikacija kopije podatka u kešu dovešće do nekonzistentnosti (neusaglasenosti) memorijskog sistema.

Keš koherencija

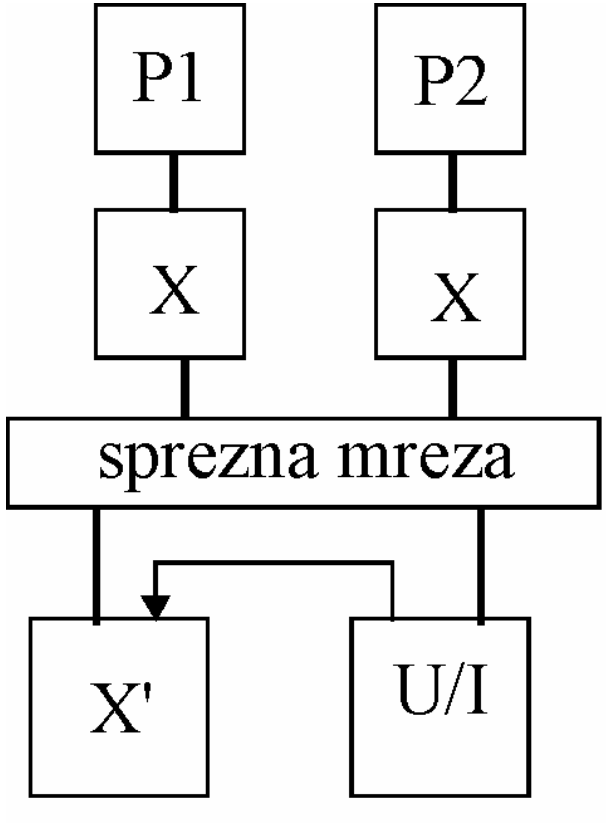
- Neka multiprocesorski sistem ima dva procesora, svaki sa privatnim kešom, koji imaju kopiju istog podatka X
  - ako jedan od procesora, recimo P1, modikuje sadržaj lokacije X na X' u svom kešu, ista vrednost biće automatski upisana u deljivu memoriju ako se koristi write through (W-T) politika za ažuriranje glavne memorije
    - kod ove politike svaki upis u keš automatski modikuje i sadržaj glavne memorije.
    - U ovom slučaju nekonzistentnost postoji izmedju dve kopije keša
  - Ako se koristi write-back (W-B) politika kod ažuriranja glavne memorije, onda će i sadrzaj glavne memorije biti nekonzistentan.
    - Kod W-B politike glavna memorija se ažurira tek kod zamene bloka



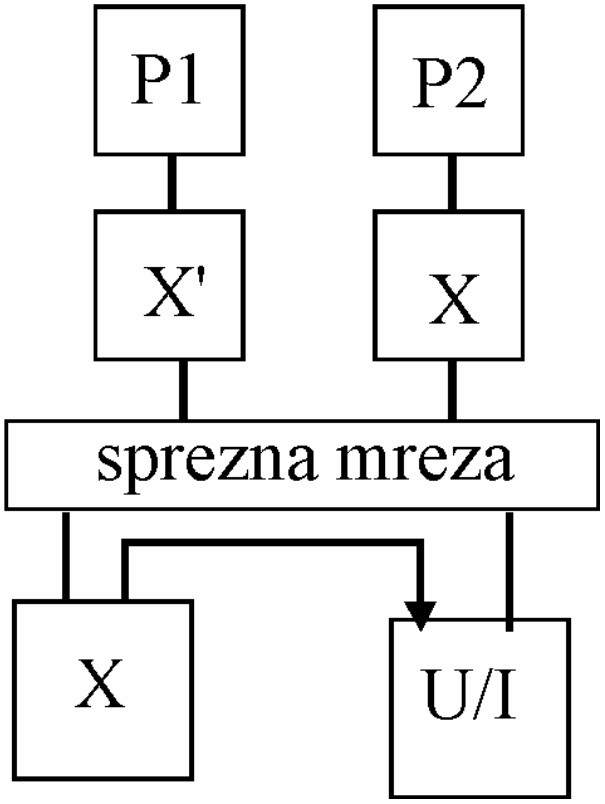
keš koherencija (nast.)

- Do nekonzistentnosti može doći i zbog U/I aktivnosti i migracije procesa u sistemima sa privatnim keš memorijama.
  - kod većine multiprocesorskih sistema organizovanih oko zajedničke magistrale, U/I procesor je takodje spregnut na magistralu.
  - U slučaju W-T politike kod ažuriranja glavne memorije do nekonzistentnoski može doći kada U/I procesor puni glavnu memoriju,
  - U slucaju W-B politike kod ažuriranja glavne memorije do nekonzistentnoski može doći kada se podaci direktno šalju iz memorije u U/I procesor

Keš koherencija-U/I aktivnosti



a)



b)

Keš koherencija

- Neki sistemi dozvoljavaju migraciju procesa, tj. da procesi budu rasporedjeni različitim procesorima u toku svog života da bi se izbalansiralo opterećenje izmedju procesora.
  - Ako se ova mogućnost koristi u sprezi sa privatnim keš memorijama takodje moze doci do nekonzistentnosti podataka.
    - Npr. proces A koji se izvrsava na procesoru P1 moze promeniti podatke usvom kešu pre nego što bude suspendovan.
    - Ako kasnije proces A migrira na procesor P2 pre nego što glavna memorija bude ažurirana, proces A moze uzeti ustajalu vrednost iz memorije.
    - Evidentno je da se samo W-T politikom kod ažuriranja glavne memorije neće održati konzistentnost memorijskog sistema, jer se ovom tehnikom ne žuriraju kopije podataka u drugim keševima.
- Postoji više prilaza za rešavanje ovog problema
  - hardverski implementirani protokoli za postizanje keš koherencije,
  - softverske tehnike.

Softverske tehnike

- Nekonzistentnost memorijskog sistema se može spreciti tako sto se nece "keširati" deljivi podaci koji se mogu menjati upisom,
  - instrukcije i drugi podaci se mogu kopirati u keš.
  - To znaci da se podaci na neki nacin moraju oznaciti.
    - To moze uciniti korisnik korišćenjem visih programskih jezika kao sto su Ada, Modula 2, Concurrent Pascal, itd., deklarišući podatke kao deljive (shared) ili nedeljive (local).
    - Alternativno, multiprocesorski kompajler može automatski klasikovati podatke kao deljive ili ne.
- Nedostatak ove tehnike je netransparentnost multiprocesorske arhitekture za korisnika ili kompajler.
  - Efikasnost ovog pristupa zavisi od mogucnosti jezika da specificira podatke kao deljive ili ne, ili od kompajlera da detektuje takve podatke.
- Pošto u prakticnim implementacijama čitava stranica mora biti deklarisana kao "cachable" ili ne, može doći do interne fragmentacije memorije, ili će se zabraniti kešovanje i onih podataka koji se mogu kopirati u keš.

Hardverski protokoli keš koherencije

- Hardverskim mehanizmima detektuju se uslovi koji mogu dovesti do nekonzistentnosti keš memorija i shodno tome obavljaju akcije koje održavju koherentnost memorijskog sistema.
- U ovu grupu protokola sadaju:
  - snoopy keš protokoli (njuškala)
  - direktorijumske šeme

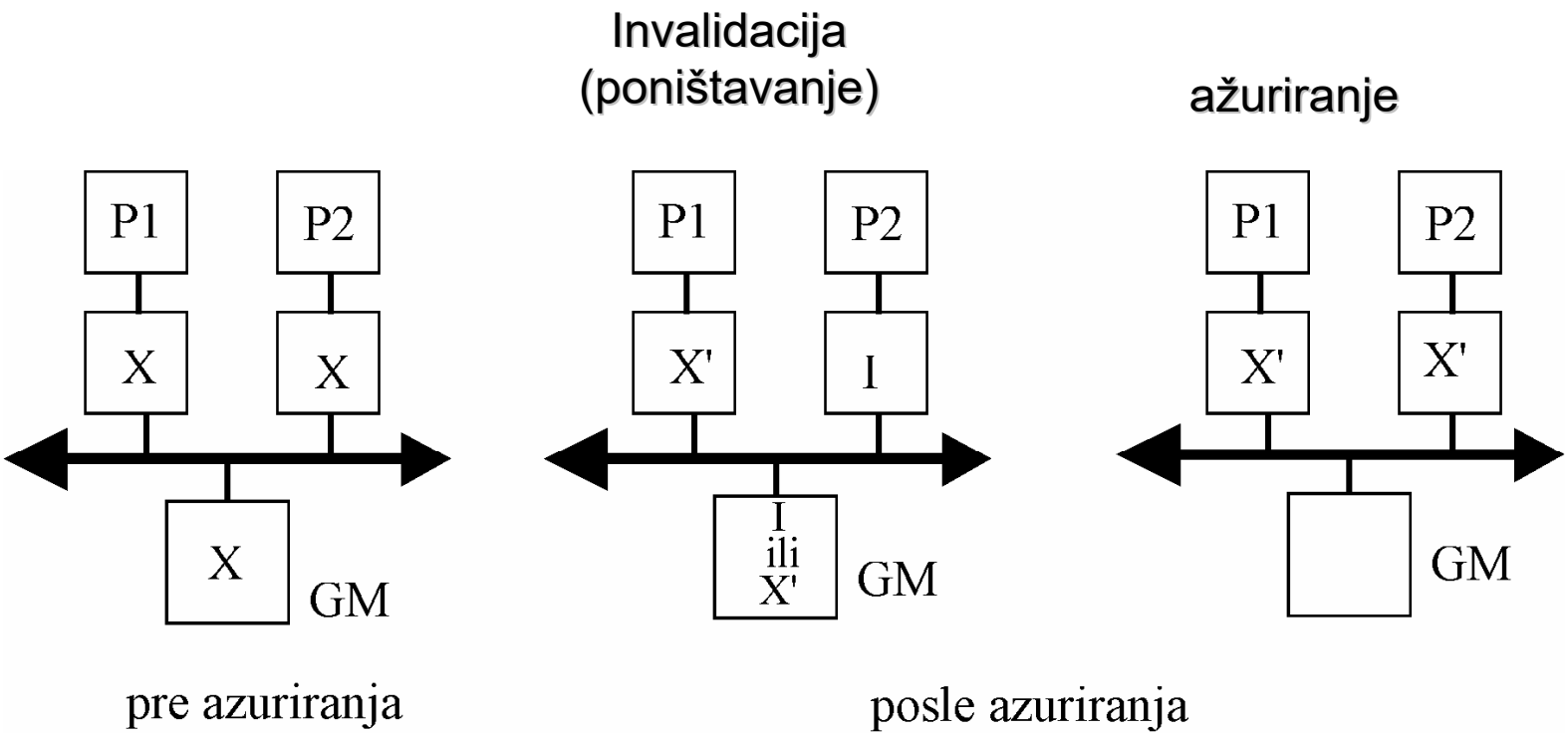
Snoopy keš protokoli

- Mnogi današnji komercijalno raspoloživi multiprocesori koriste memorijske sisteme organizovane oko zajedničke magistrale.
  - Magistrala je pogodna za odrzavanje keš koherencije jer kao jedinstveni sprežni put, dozvoljava da svi procesori u sistemu nadgledaju memorijske transakcije.
  - Ako transakcija na magistrali ugrožava konzistentnost podataka u lokalnom kešu, specijalni hardver (keš kontroler) može preduzeti akcije da poništi lokalnu kopiju (tj. proglasi je nevažećom).
  - Protokoli koji koriste ovaj mehanizam za postizanje koherentnosti zovu se snoopy (njuškala) protokoli jer svaki keš kontroler "njuška" transakcije drugih keševa.

Snoopy keš protokoli

- Praktikuju se dva prilaza (politike) za postizanje keš koherencije:
  - 1. invalidacija (poništavanje) pri upisu (write-invalidate)
  - 2. ažuriranje pri upisu (write-update)
- Kod politike invalidacije pri upisu konzistentnost većeg broja kopija održava se na sledeci nacin:
  - Kada procesor ažurira svoj lokalni blok podataka sve ostale kopije tog bloka u sistemu se poništavaju.
  - Sva naredna ažuriranja od strane procesora koji je izvšio invalidaciju se izvode lokalno (bez izdavanja komande invalidacije) jer u sistemu postoji samo jedna važeća kopija.

Snoopy keš protokoli



"Write-Once" protokol

- "Write-Once" -prvi protokol ovog tipa (1983.god.)
  - Koristi W-T politiku kod ažuriranja glavne memorije pri prvoj modikaciji keš bloka.
  - Nakon prvog upisa deljiva memorija se ažurira korišćenjem W-B politike
- Protokol razlikje četiri stanja kopije keš bloka:
  - Valid (važeća): U kešu je kopija koja je konzistentna (jednaka) sa memorijskom kopijom i nije modikovana (tj. konzistentna je sa ostalim kopijama, ako postoje, u drugim keševima).
  - Invalid: nevažeća kopija (tj. kopija je poništena)
  - Reserved (rezervisano): podatak je u kešu modikovan samo jednom i keš kopija je konzistentna sa kopijom u glavnoj memoriji, koja je jedina druga (sve ostale kopije, ako su postojale su invalidirane)
  - Dirty (prljava): keš jemodikovan više puta i kopija je jedinstvena u sistemu.
    - U sistemu moze postojati samo jedna prljava kopija.

Write-once protokol (nast.)

- Da bi se održala konzistentnost memorijskog sistema protokol koristi dva skupa komandi:
  - Komande koje izdaje procesor lokalnom kešu (lokalne procesorske komande)
    - P_Read (komanda za čitanje lokalnog keša),
    - P_Write (komanda za upis u lokalni keš).
  - Komande konzistencije koje se izdaju preko zajedničke magistrale:
    - Read_Blk (čitanje bloka iz memorije),
    - Write_Blk (upis bloka u memoriju),
    - Write_Inv (modifikuje lokalnu kopiju i invalidira sve ostale kopije),
    - Read_Inv (čita blok, modifikuje ga i invalidira sve ostale kopije).
- Kod obraćanja procesora lokalnom kešu mogu nastupiti sledeći dogadjaji:
  - pogodak kod čitanja
  - promašaj kod čitanja
  - pogodak pri upisu
  - promašaj pri upisu
  - zamena bloka

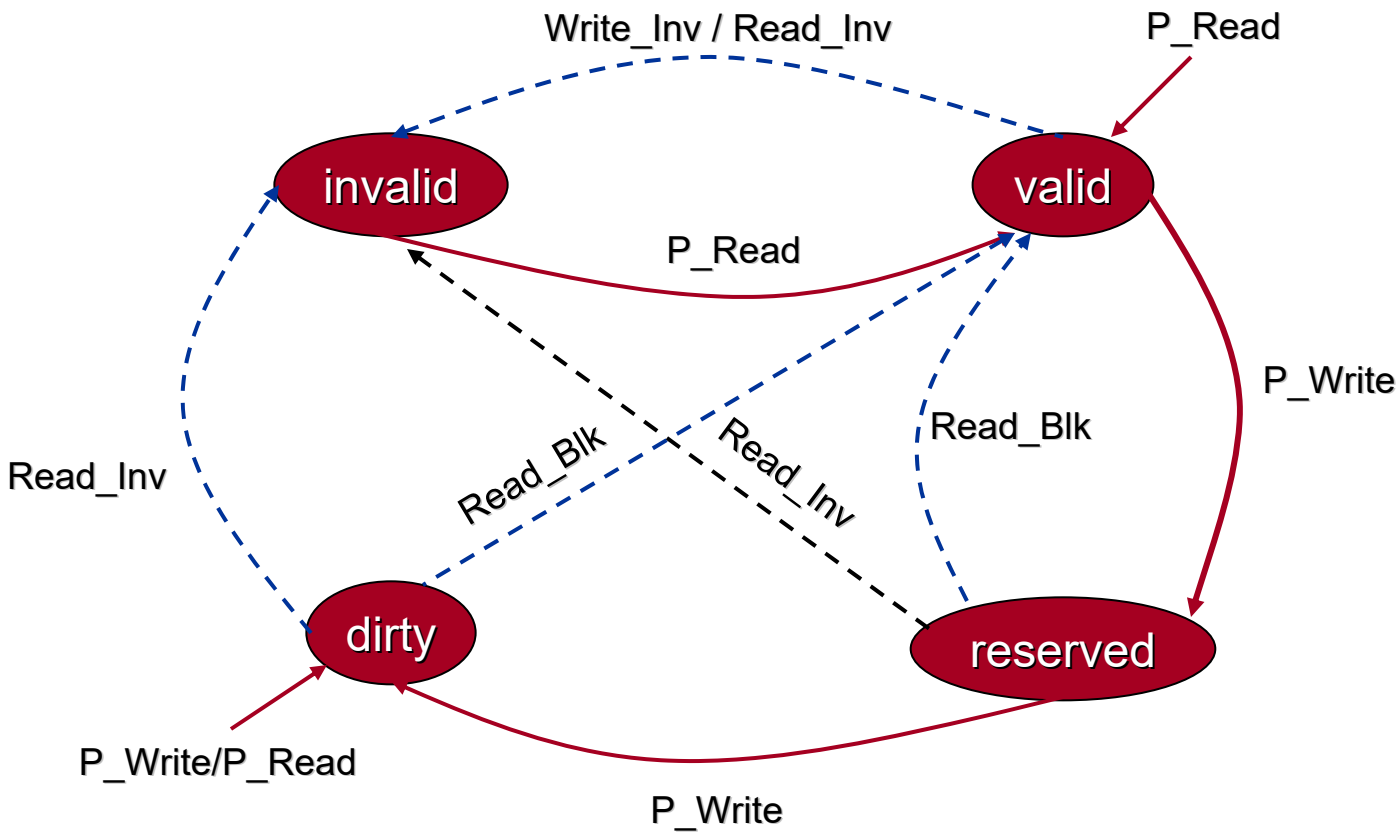
Write-once protokol (nast.)

- pogodak kod čitanja:
  - obavlja se lokalno i ne uslovljava promenu stanja keš kopije i ne zahteva izdavanje komandi konzistencije
- promašaj kod čitanja :
  - ako nastupi promasaj kod čitanja i ni jedan keš u sistemu ne sadrži prljavu (dirty) kopiju, inicira se Read_Blk operacija i podatak (tj. ceo blok) se prenosi iz glavne memorije u lokalni keš.
  - ako postoji prljava kopija, tada će odgovarajuci keš kontroler zabraniti čitanje glavne memorije i poslati kopiju kešu koji je izdao zahtev:
    - obe kopije postaju validne i ažurira se i glavna memorija.
- pogodak pri upisu:
  - ako nastupi pogodak pri upisu i kopija je u stanju reserved ili dirty novo stanje kopije je dirty.
  - ako je kopija bila u stanju valid , tada se svim keševima šalje komanda Write_Inv pomoću koje se invalidiraju sve kopije.
  - Memorijska kopija se ažurira, novo stanje lokalne keš kopije je reserved.

Write-once protokol (nast.)

- promašaj pri upisu:
  - kada nastupi promašaj pri upisu procesor mora prvo pribaviti kopiju iz glavne memorije ili iz keša koji ima dirty kopiju .
    - Ovo se postiže slanjem Read_Inv komande koja će pribaviti i ažurirati blok i invalidirati sve ostale keš kopije.
  - lokalna kopija nakon ovoga se nalazi u stanju dirty.
- zamena bloka:
  - ako je stanje bloka dirty, kod zamene bloka se vrši upis u glavnu memoriju (Write_Blk), a ako je blok invalid, reserved ili valid nema upisa u memoriju kod zamene bloka.

Write-once protokol (nast.)



- pune linije – komande koje izdaje lokalni procesor
- isprekidane linije – komande koje izdaje udaljeni procesor preko sistemske magistrale

Snoopy protokol sa ažuriranjem pri upisu

- Firefly protokol
- postoje tri moguća stanja keš kopije:
  - Valid-exclusive
    - postoji samo jedna keš kopija i ona je konzistentna sa glavnom memorijom.
  - Shared
    - postoji više kopija i sve su konzistentne
  - Dirty
    - postoji samo jedna kopija i ona nije konzistentna sa glavnom memorijom.
- Firefly protokol koristi write-back (W-B) politiku za ažuriranje privatnih blokova (blokovi koji su u stanju Valid-exclusive i Dirty), a W-T za deljive (shared) blokove.
  - Da li je blok deljiv ili privatni, odredjuje se u toku izvršenja programa.
  - Da bi se održala konzistentnost kod modifikacije deljivog (shared) bloka koristi se Write_Update komanda koja ažurira sve kopije u sistemu.

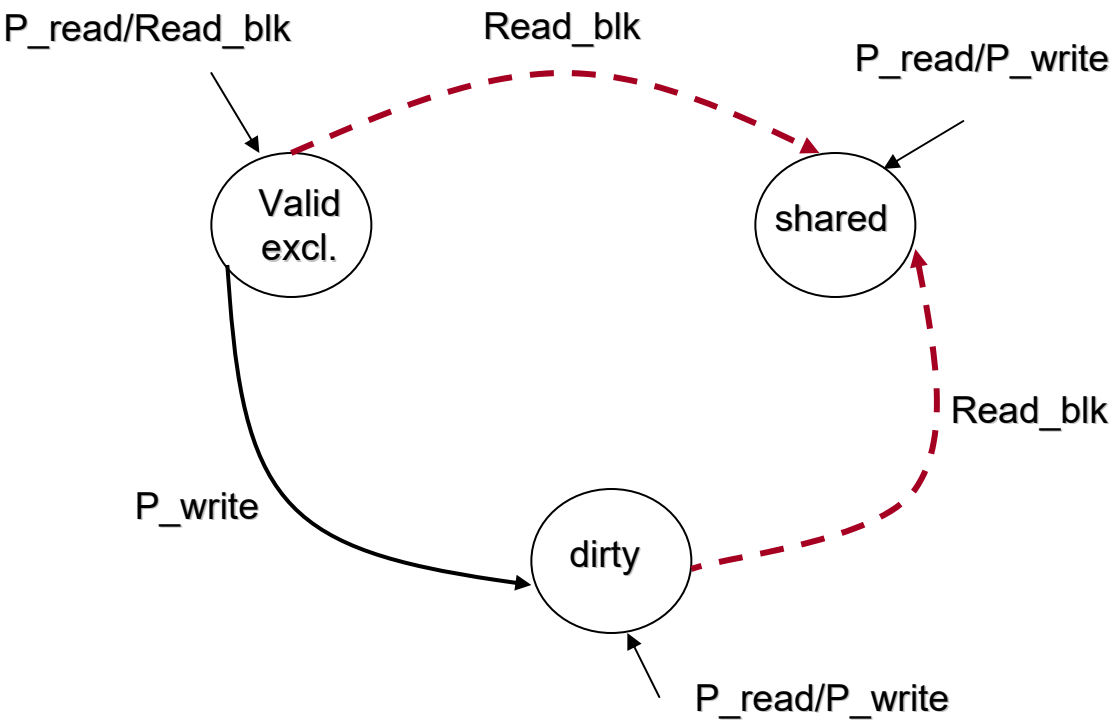
Firefly protokol

- U zavisnosti koji je dogadjaj nastupio kod obraćanja kešu preduzimaju se sledeće akcije:
  - pogodak kod čitanja:
    - obavlja se lokalno i ne uslovljava promenu stanja keš kopije i ne zahteva izdavanje komandi konzistencije
  - promašaj kod čitanja:
    - ako postoji Shared ili Valid-Exclusive tada se blok pribavlja iz glavne memorije.
    - Ako postoji dirty kopija tada keš sa dirty kopijom predaje blok podataka, ažurira glavnu memoriju, a rezultujuće stanje kopije je Shared.
    - Ako ni jedan keš ne poseduje kopiju bloka, tada se on pribavlja iz glavne memorije a stanje kopije je Valid-exclusive.
  - pogodak pri upisu:
    - ako je blok Dirty ili Valid-exclusive tada se upis vrši lokalno i novo stanje je Dirty.
    - Ako jekopija Shared sve ostale kopije bloka (ukljucujuci i memorijsku) se ažuriraju.

Firefly protokol

- promašaj pri upisu:
  - kopija dolazi iz drugih keševa ili iz glavne memorije
  - ako dolazi iz memorije stanje kopije nakon upisa je dirty.
  - U ostalim slucajevima sve kopije se ažuriraju a novo stanje je shared.
- zamena bloka:
  - ako je stanje bloka dirty, kod zamene bloka se vrši upis u glavnu memoriju (Write-Blk).
  - U ostalim slucajevima akcije se ne preduzimaju.

Firefly - promene stanja keš bloka



Invalidacija naspram ažuriranja pri upisu

	Write-invalidate	Write-update
Upis	Procesor koji modifikuje svoj lokalni keš primorava sve ostale da ponište svoju kopiju	Procesor koji modifikuje svoj lokalni keš primorava sve ostale da ažuriraju svoju kopiju
Prednost	Manji saobraćaj na magistrali	Ostali procesori brže mogu dobiti važeće podatke
Nedostatak	Sporije ažuriranje podataka u kešu	Veći saobraćaj na magistrali

Snoopy protokoli - zaključak

- Pogodni za sisteme sa malim brojem procesora
- Jednostavna implementacija
- Nisu skalabilni

Direktorijumske šeme

- Kada se koriste vešestepene sprežne mreže za gradnju multiprocesorskih sistema sa velikim brojem procesora, snoopy protokoli se moraju modikovati da bi se prilagodili mogućnostima mreže.
  - posto je broadcast (emisiju) veoma skupo obaviti kod višestepenih mreža, komande konzistentnosti se šalju samo onim keševima koji imaju kopiju bloka.
  - Da bi se to učinilo potrebne su tačne informacije o tome koje keš memorije imaju kopiju odredjenog bloka.
  - Ove informacije se smeštaju u direktorijumima (adresarima).
- protokoli koji koriste informacije zapamćene u direktorijumima za održavanje koherentnosti memorijskog sistema zovu se direktorijumske šeme.

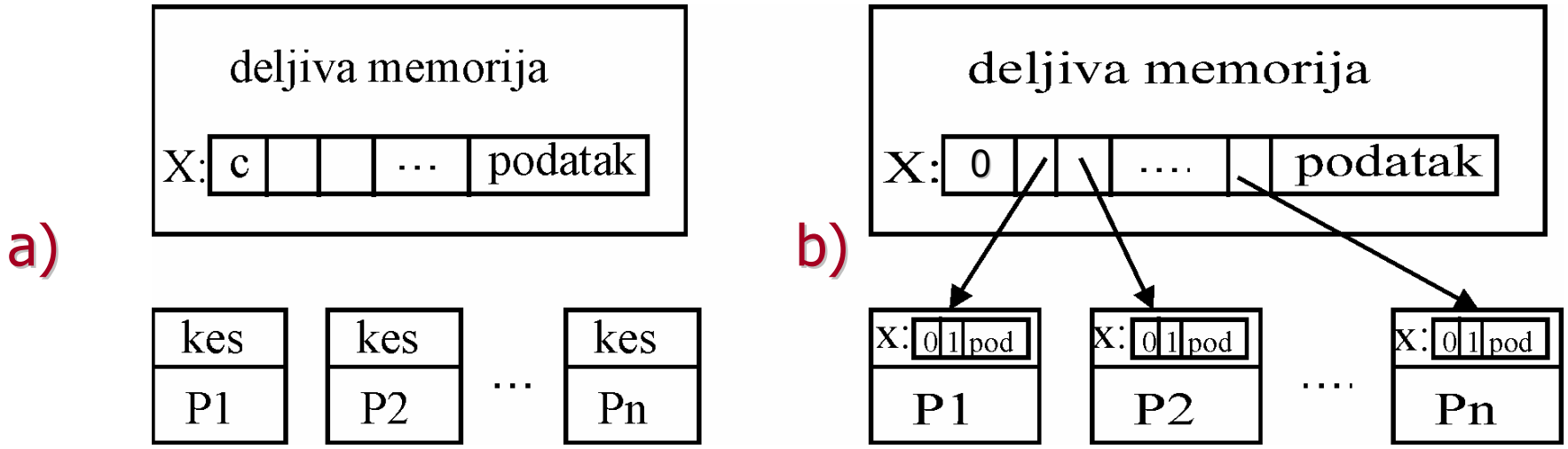
Direktorijumske šeme

- Svaki memorijski modul sadrži poseban direktorijum koji beleži stanje i prisustvo memorijskog bloka u odredjenom kešu.
- Svaki direktorijumski ulaz (linija) sadrži
  - veći broj pokazivača koji ukazuju na keš koji sadrži kopiju bloka i
  - jedan "dirty" bit kojim se ukazuje kada odredjeni keš ima, ili ne, dozvolu da modikuje podatak..
- Postoje tri tipa direktorijumskih protokola:
  - protokoli sa potpuno preslikanim adresarima (Full-map directories)
  - protokoli sa ograničenim adresarima (limited directories)
  - protokoli sa ulančanim adresarima (chained directories)

Protokoli sa potpuno preslikanim adresarima

- Svaki direktorijumski ulaz u glavnoj memoriji sadrži po jedan bit za svaki procesor u sistemu i jedan "dirty" bit.
  - Svaki bit pridružen procesoru ukazuje na status bloka u odgovarajućem procesorskom kešu (prisutan ili ne).
  - ako je "dirty" bit postavljen, tada je jedan i samo jedan procesorski bit postavljen i taj procesor ima pravo da vrši upis u kešovani blok (tj. da ga modikuje).
- Svaki keš direktorijum sadrži dva bita stanja za svaki kešovani blok.
  - Jedan bit ukazuje da li je blok važeći, a
  - drugi da li se u važeći blok moze vršiti upis.
- Keš koherentni protokol mora držati bitove stanja u memorijskom direktorijumu i u kešu konzistentnim.

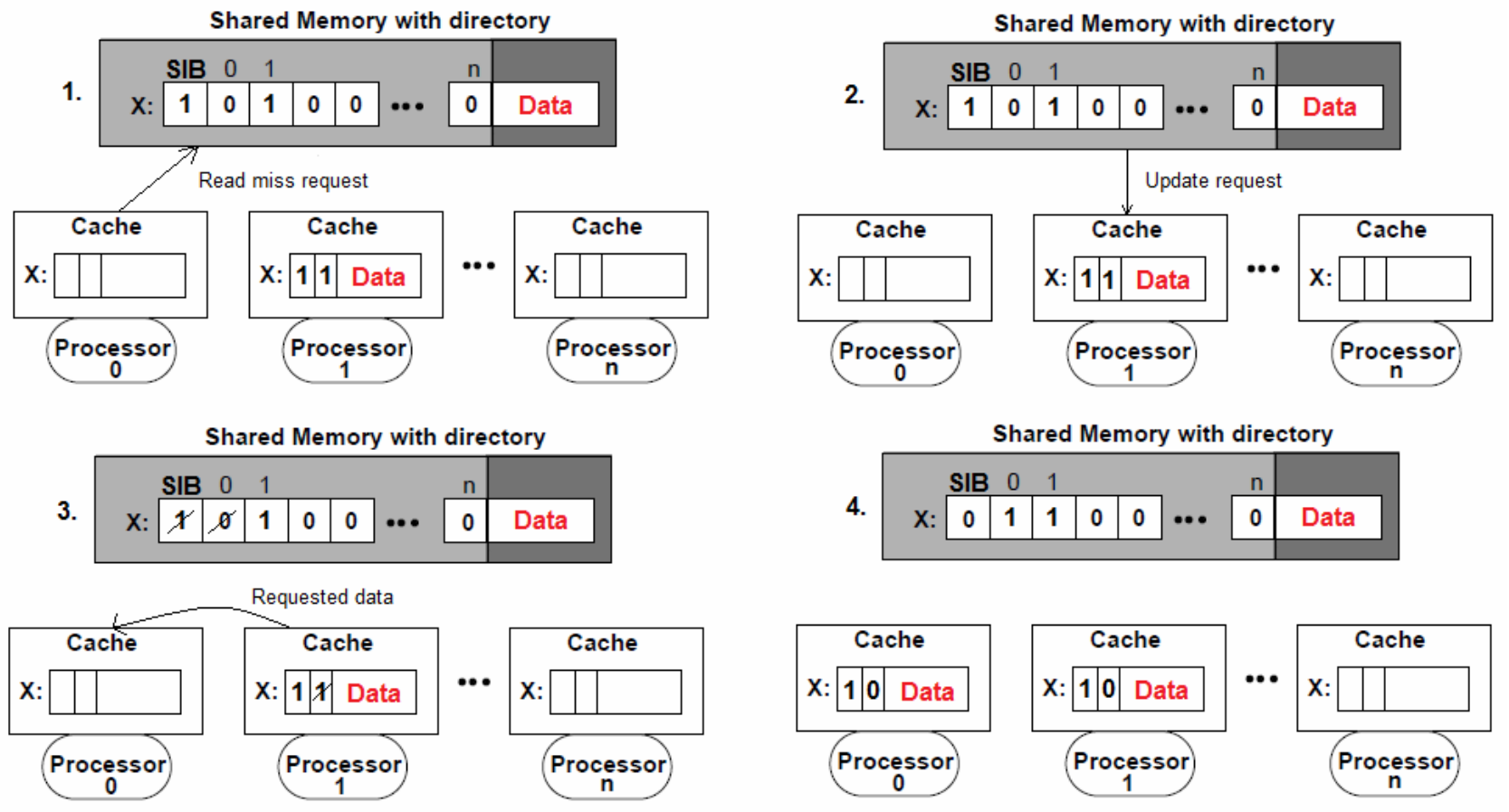
Protokoli sa potpuno preslikanim adresarima



- Sl a) ni jedan keš ne sadrži kopiju lokacije X.
- Sl b) procesori P1, P2 i Pn zahtevali kopiju podatka X.
  - tri bita u odgovarajućem adresarskom ulazu u glavnoj memoriji koji ukazuju na prisustvo bloka u odredjenom kešu se postavljaju.
  - u keš adresaru se postavljaju odgovarajući bitovi stanja da ukažu na važecu kopiju.
  - U adresaru u glavnoj memoriji "dirty" bit je obrisan (c), i ukazuje da ni jedan procesor nema pravo da modikuje kopiju podataka u svom kešu

Full Map Directory

- Read miss (dirty bit = 1)

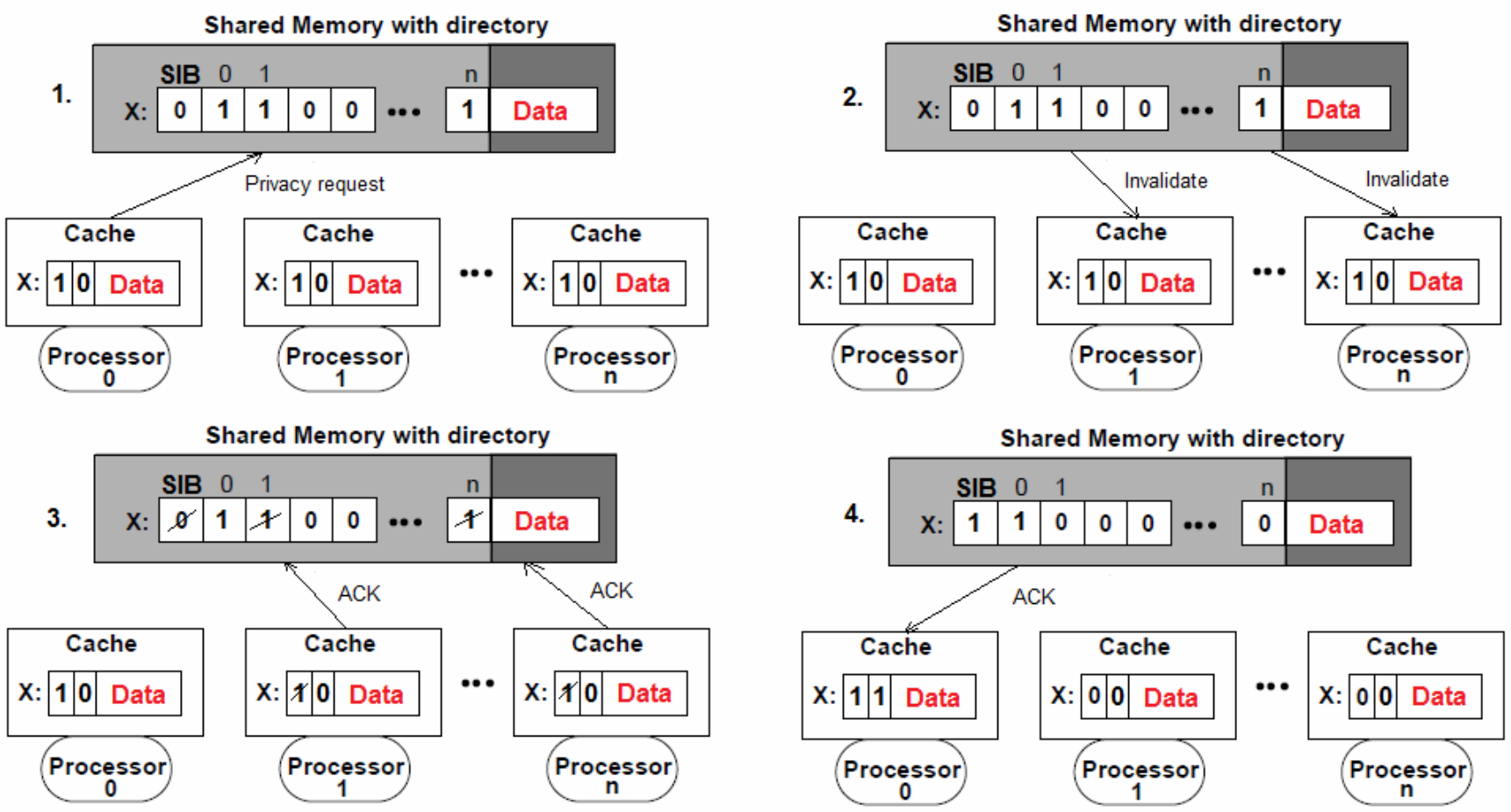


Protokoli sa potpuno preslikanim adresarima

- Šta se dešava ako jedan procesor, recimo Pn, izda komandu za upis u svoj keš (Cn):
  1. Keš Cn detektuje da je blok koji sarži lokaciju X važeci, ali da procesor nema pravo upisa.
  2. Keš kontroler Cn upućuje zahtev za modikacijom odgovarajućem memorijskom modulu koji sadrži lokaciju X i zaustavlja svoj procesor, Pn.
  3. Memorijski modul izdaje zahtev za invalidacijom keševima C1 i C2.
  4. Keševi C1 i C2 primaju invalidacione zahteve, postavljaju odgovarajuce bitove koji ukazuju da je blok koji sadrži lokaciu X nevažeci i vraćaju potvrdu o priznavanju zahteva nazad memorijskom modulu.
  5. Memorijski modul prima potvrdu o invalidaciji, postavlja "dirty" bit, briše pokazivače na keševe C1 i C2 i predaje dozvolu za upis kešu Cn.
  6. Keš Cn prima poruku o dozvoli upisa, ažurira stanje u kešu i aktivira procesor Pn.
- Ovakvim redosledom akcija protokol garantuje da će memorijski sistem ostati konzistentan.

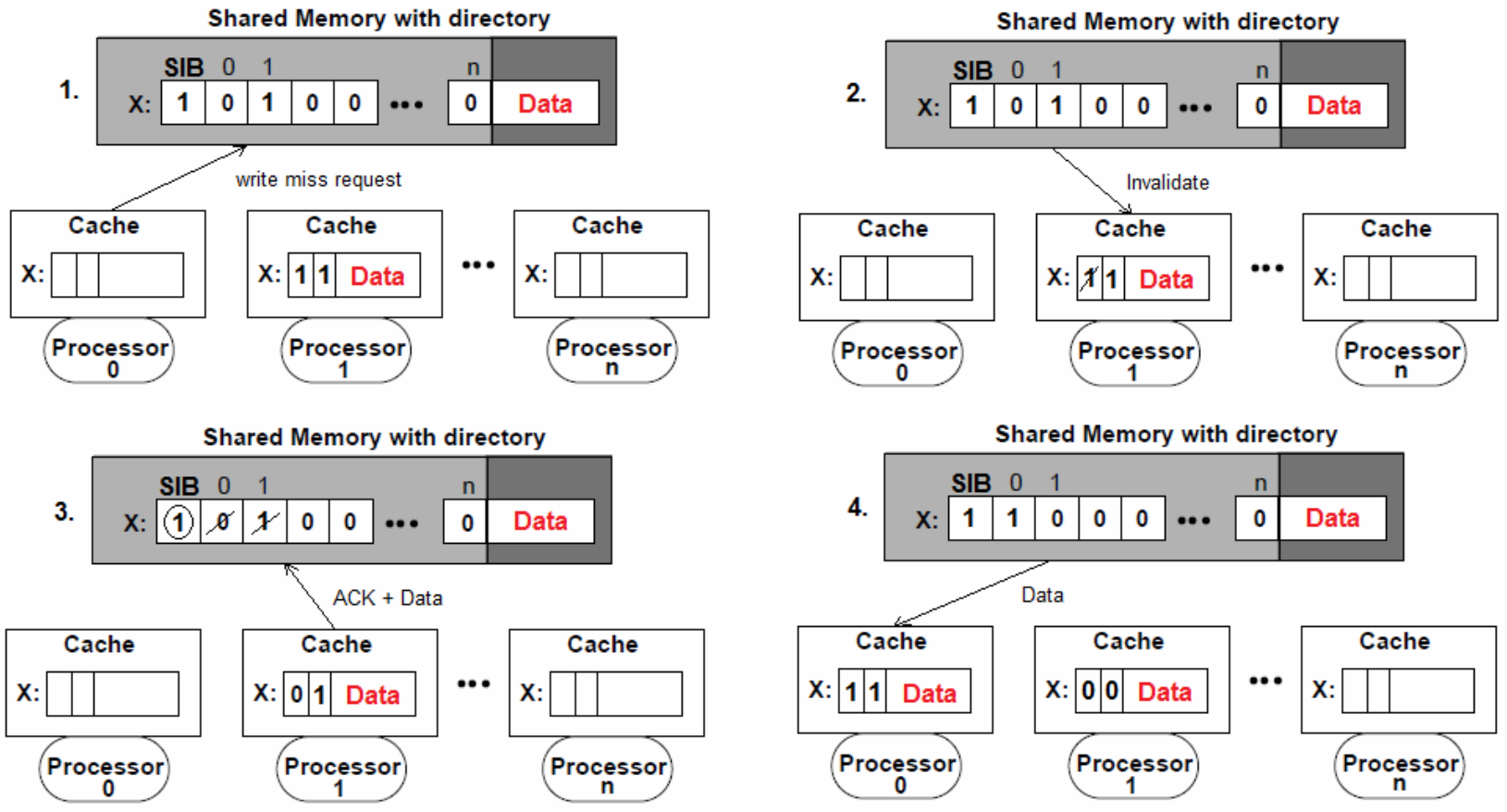
Full Map Directory

- Write hit



Full Map Directory

- Write miss



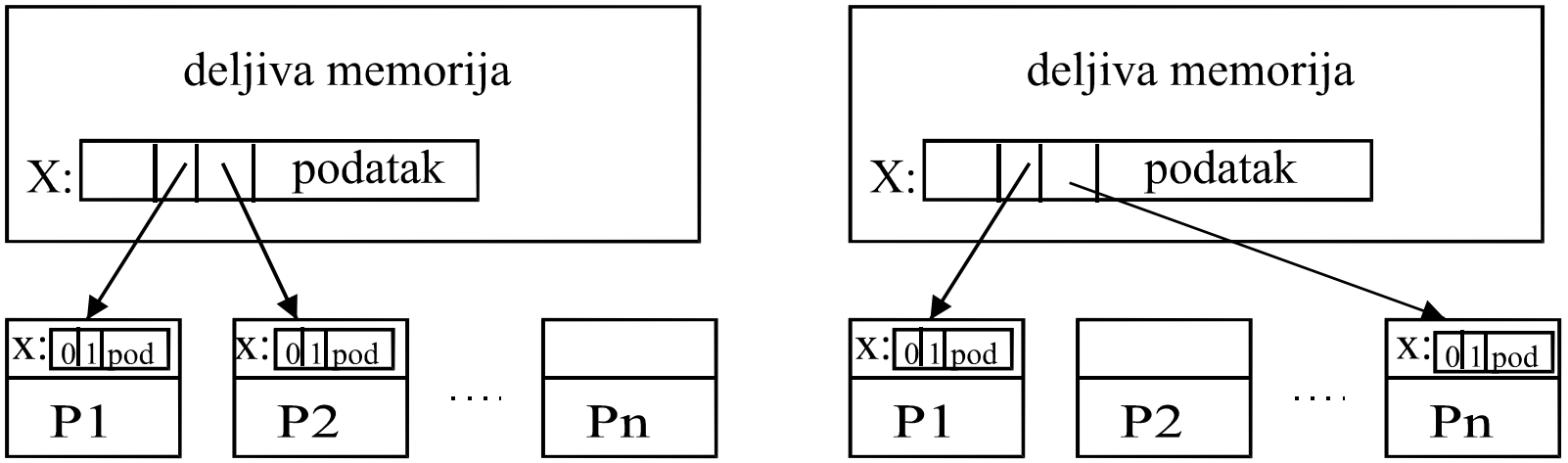
Protokoli sa potpuno preslikanim adresarima

- Nedostaci:
  - za svaki memorijski blok u adresaru zahteva se po jedan bit za svaki procesor.
  - Ako ima N procesora svaki direktorijumski ulaz ima N pokazivača.
  - Sa porastom broja procesora u sistemu, proporcionalno se povećava i adresarski prostor (tj gubitak memorije zbog vodjenja evidencije u adresarima).
  - Isto važi i ako se povećava veličina memerije.
  - Ovi problemi se mogu resiti ako sekoriste ograničeni adrsari.

Protokoli sa ograničenim adresarima

- Ograničava se broj jednovremenih kopija jednog memorijskog bloka (tj. samo odredjeni broj procesora moze sadržati kopiju bloka u svom kešu).
  - Sličan je protokolu sa potpuno preslikanim adresarima, izuzev kada više keševa nego što je dozvoljeno zahteva kopiju jednog bloka podataka.
  - Svaki direktorijumski ulaz u glavnoj memoriji sadrži odredjeni (ograničeni) broj pointera na procesore koji sadrže kopiju bloka i jedan "dirty" bit koji ukazujeda li se blok može modikovati.
- Primer
  - ako je broj kopija ograničen na dve, i ako se kopije nalaze u keševima procesora P1 i P2,
  - tada će direktorijumski ulazi u adresaru glavne memorije sadržati pointere na procesore P1 i P2
  - Ako procesor Pn zahteva kopiju istog bloka podataka, kontroler glavne memorije mora invalidirati kopiju bloka u kešu procesora P1 ili P2 i zameniti pokazivač

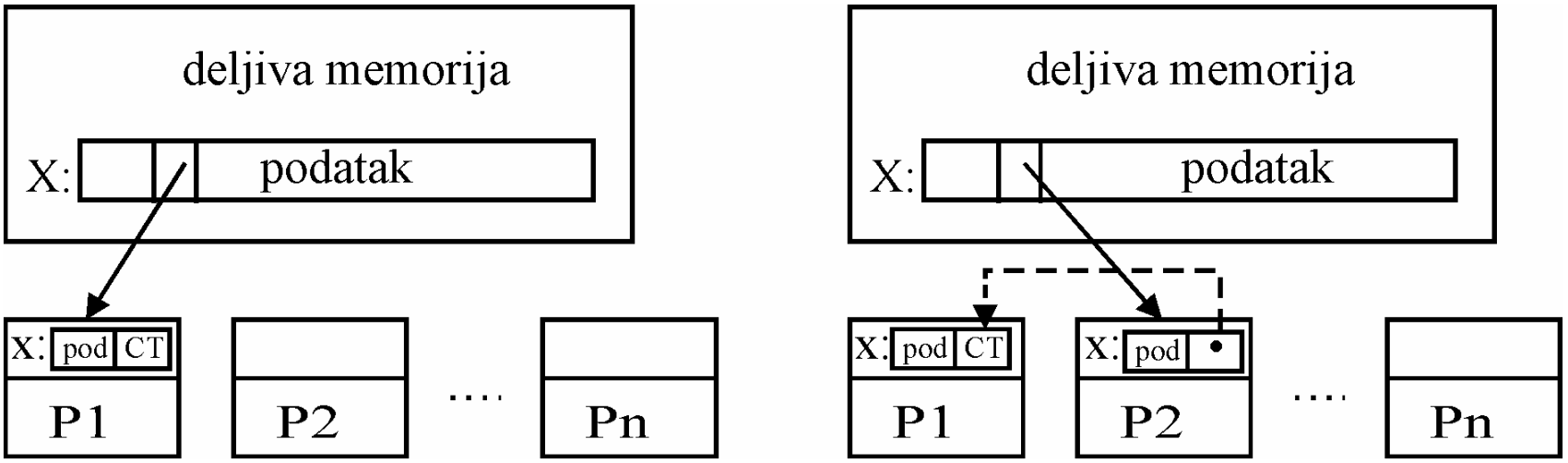
Protokoli sa ograničenim adresarima



Protokoli sa ulančanim adresarima

- Dozvoljavaju proširljivost sistema bez ograničavanja broja kopija jednog memorijskog bloka.
  - Ovaj tip adresarske šeme zove se 'ulančani adresari" jer čuva trag o kopijama bloka preko lanca adresarskih pokazivaca.
  - Formiranje adresarskog lanca :
    - Pretpostavimo da u sistemu ne postoji kopija bloka sa lokacijom X.
    - ako P1 čita lokaciju X, glavna memorija šalje kopiju bloka kešu procesora P1 i ujedno pokazivač na kraj lanca CT (Chain Termination).
    - U u adrsaru glavne memorije čuva se pokazivač na procesor P1
    - Ako P2 zahteva čitanje lokacije X, glavna memorija šalje kopiju bloka kešu procesora P2 i pokazivac na P1.
    - Sada se u adrsaru glavne memorije čuva se pokazivač na procesor P2
    - Ponavljanjem ovih koraka svi keševi mogu dobiti kopiju bloka.

Protokoli sa ulančanim adresarima



- Neka jedan od procesora, recimo P3, zahteva da modikuje sadržaj lokacije X.
  - Da bi se održala konzistentnost memorijskog sistema potrebno je da se komanda invalidacije prosledi kroz lanac.
  - Kontroler glavne memorije ne daje dozvolu modikacije procesoru P3 sve dok ne primi potvrdu o invalidaciji od procesora koji sadrži CT pointer.
  - S obzirom da se informacija o invalidaciji prenosi od jednog do drugog kesa u lancu, protokol se zove i "gossip" (tračarenje) protokol.