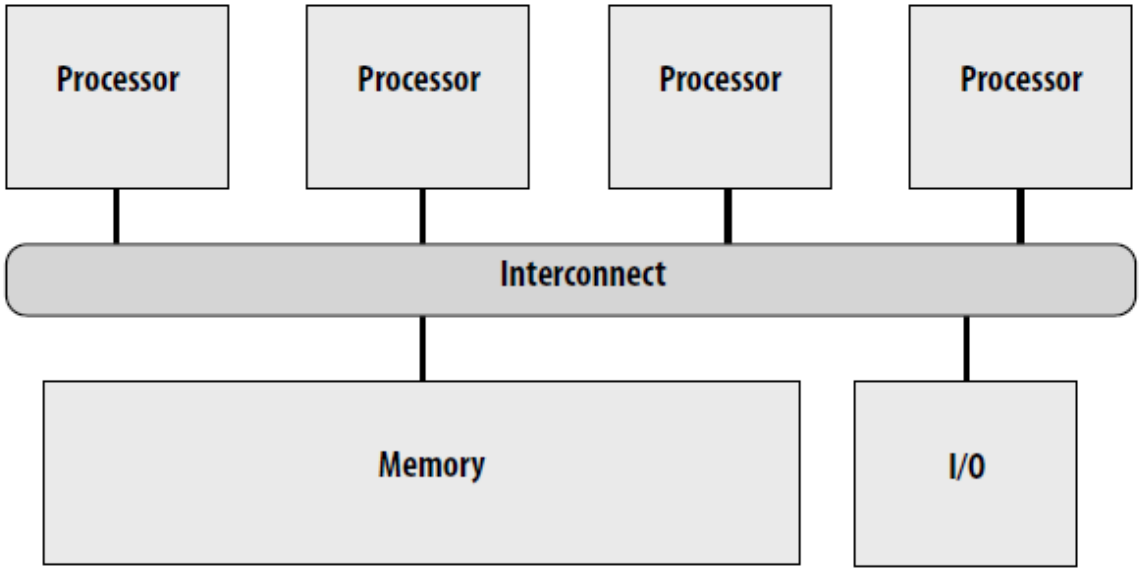
Multiprocesorski sistemi

Multiprocesorski sistemi sa zajedničkom memorijom

Milo Tomašević

SI4MPS

Sistemi sa zajedničkom memorijom



- Programski model SM direktno podržan u HW
  - Čitanja i upisi deljenih virtuelnih adresa hardverski podržani
  - Za aplikaciju izgleda kao *multitasking* jednoprocesorki sistem
  - Može i slanje poruka, obično u SW (*run-time* biblioteke, bez OS!)

Sistemi sa zajedničkom memorijom

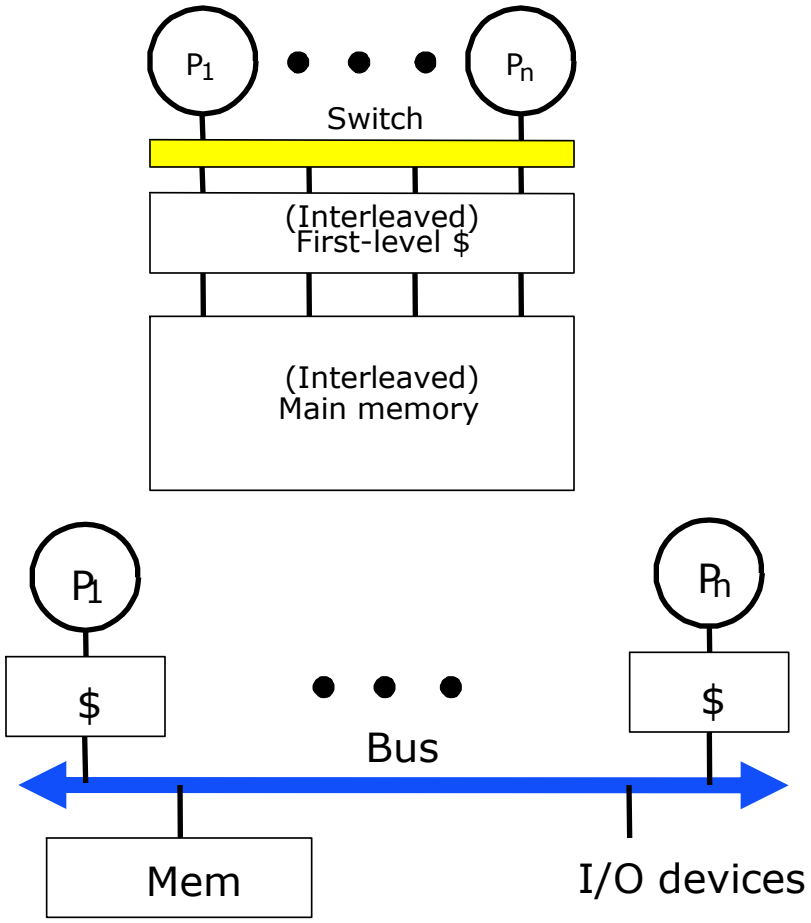
- Pogodni i za multiprogramiranje i za paralelizaciju
  - Efikasna deljivost resursa (procesori, memorije)
  - Uobičajeni uniforman pristup podacima (L/S)
  - Prirodna nadgradnja OS
  - Pogodni za izvršavanje OS rutina
  - Jeftina i modularna proširljivost
- Česta forma - SMP (simetrični multiprocesor)
- Zajednički virtuelni adresni prostor + simetričan pristup procesora čitavoj memoriji
  - Dominantni na tržištu servera (srednji nivo)
  - Sve prisutniji i na desktopovima i CMP (niži nivo)
  - Gradivni blok za sisteme višeg nivoa

Sistemi sa zajedničkom memorijom

- Imenovanje & sinhronizacija
  - Implicitna komunikacija kroz deljeni prostor
  - (teža za optimizaciju)
  - Sinhronizacija pomoću operacija nad deljenim adresama
- Latencija & propusni opseg
  - Izbeći dugačke latencije pristupa podacima
  - Smanjiti potrebu za propusnim opsegom
  - Korišćenje keš memorija - automatska replikacija i prenos podataka
  - Ključni problem – organizacija proširene memorijske hijerarhije!

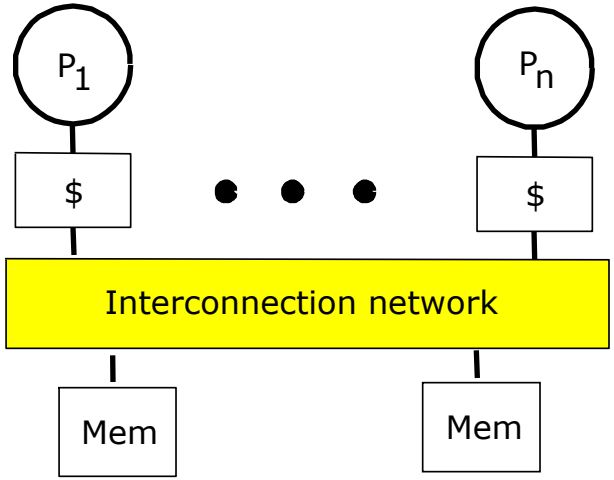
Proširene memorijske hijerarhije

- Zajednička keš memorija
  - npr., Alliant FX-8 – 8 x 68020 sa krosbarom na deljeni $M
  - npr., Encore & Sequent po dva N32032 na ploči sa $M
  - Opcija za deljeni L1 ili (često) L2 i L3 u CMP
- Zajednička magistrala
  - Preovlađuje u komercijalnim sistemima nižeg i srednjeg nivoa
  - Fleksibilnost
  - Ograničena skalabilnost

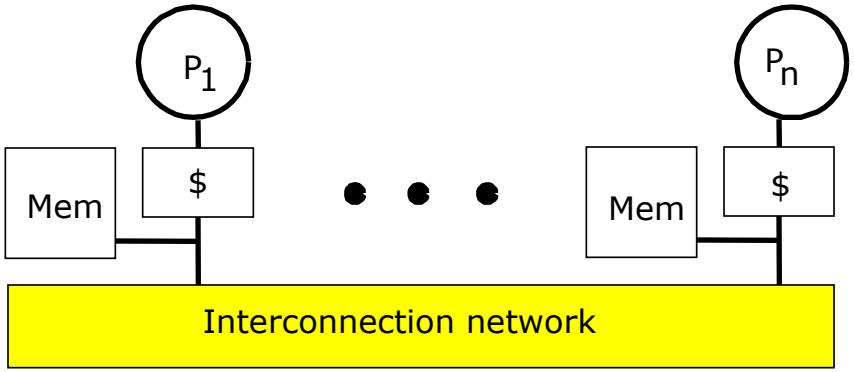


Proširene memorijske hijerarhije

- UMA
  - "*Dance-hall*"
  - Simetričnost pristupa
  - Skalabilna mreža

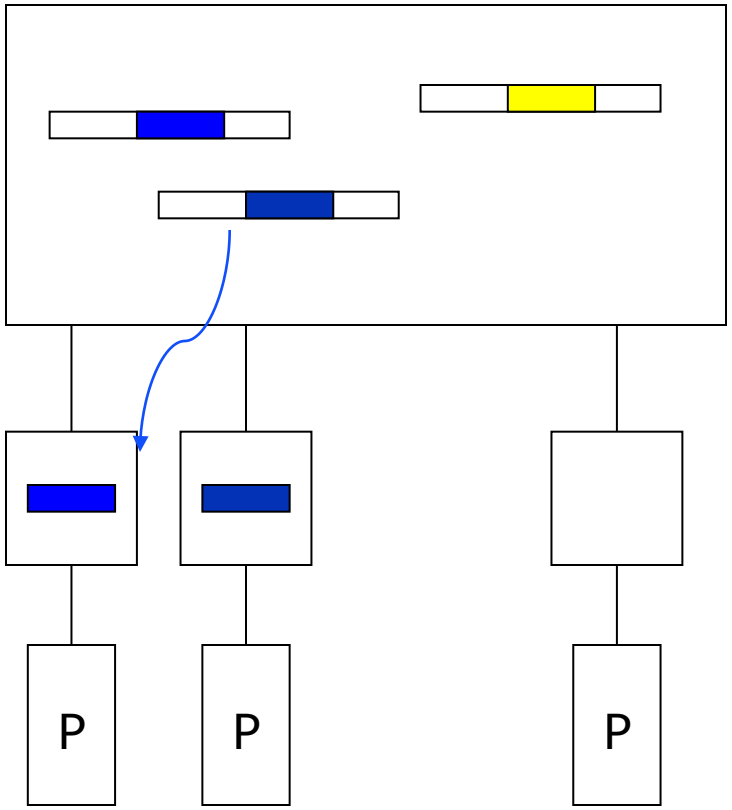


- NUMA
  - Lokalne memorije
  - Nesimetričnost pristupa
  - Klaster može imati arhitekturu sa zajedničkim kešom ili magistralom

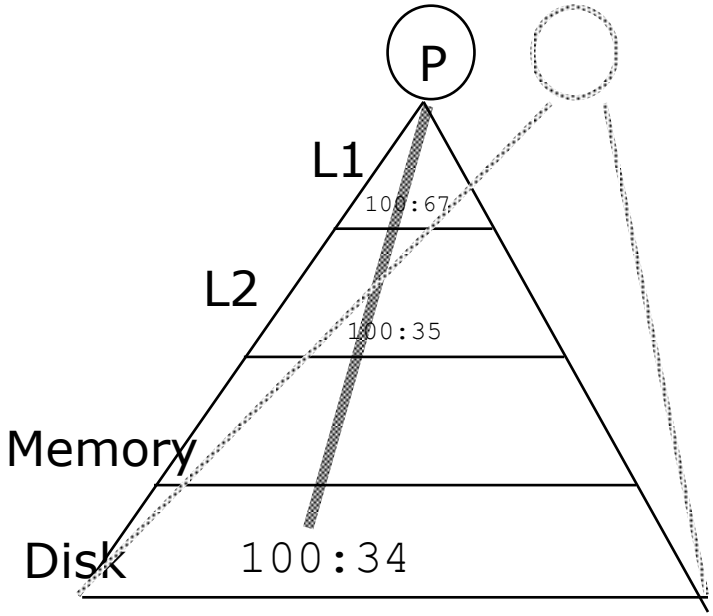


Korišćenje keš memorija

- Keširanje podataka
  - Automatska replikacija podataka blizu procesora
  - Efikasno deljenje istog podataka
  - Privatni deo hijerahije
- Esencijalno za performanse
  - Smanjena kontencija za mrežu i memoriju
  - Manja latencija pristupa (manja memorija, kraći put)
  - Manje potrebe za propusnim opsegom mreže i memorije

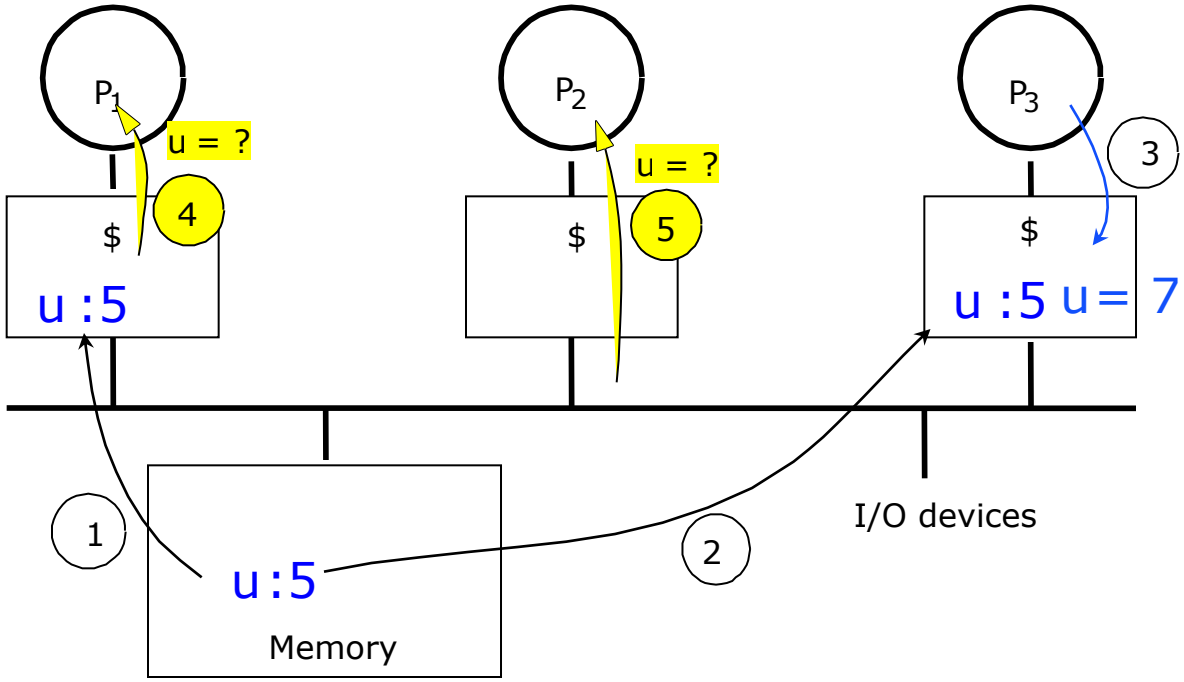


Intuitivni memorijski model



- Čitanje na nekoj adresi treba da vrati poslednju upisanu vrednost na tu adresu
- Lako ostvariti u jednoprocesorskim sistemima (osim za U/I)
- Osnova apstrakcije zajedničke memorije
- Šta ako upisi i čitanja potiču od različitih procesora !?

Problem koherencije keš memorija

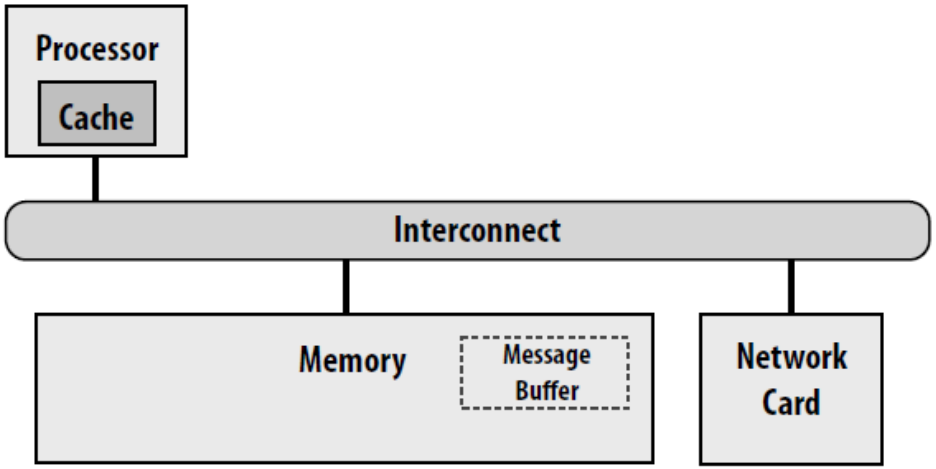


- Različiti procesori vide različite vrednosti (kroz drugu hijerarhiju)
- Apstrakcija zajedničke memorije nije implementirana u jednoj memoriji
- Problem nastaje i za WT i za WB (još izraženiji!)

Problem koherencije keš memorija

- Problem koherencije keš memorije
  - Uzrok - replikacija deljivih podataka sa mogućnošću upisa u privatnim keš memorijama
  - Može da nastane usled migracije procesa čak i za logički privatne podatke
  - Čak i u jednoprocesorskim sistemima pri U/I sa DMA
  - Pojava "lažne deljivosti" čak i za privatne podatke
  - Narušava intuitivni memorijski model
- Važnost rešavanja
  - Za korektno izvršavanje programa
  - Vrlo izražen uticaj na performanse (često se dešava)
- Kako ga rešiti?
  - Organizovati memorijsku hijerarhiju da se problem izbegne
  - Preduzeti akcije da se problem prepozna i reši

Problem koherencije kod U/I



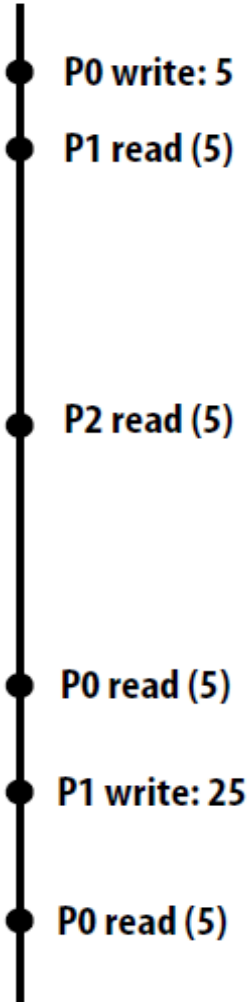
- Upisi/čitanja procesora i U/I uređaja (DMA)
  - Pristupi memorijskom baferu zaobilaze keš memoriju
- Rešenje?
  - Nekeširani upisi u bafer u kodu drajvera
  - Podrška OS (*flush* $M nakon U/I, nekeširane VM stranice koje odgovaraju baferu)
  - Manji uticaj na performanse (retko se dešava)

Koherentni memorijski sistem

- Problemi sa intuitivnim memorijskim modelom
  - Šta znači poslednji upis?
  - Šta ako su upisi od dva procesora itovremeni ili toliko bliski u vremenu da ne stigu da se propragiraju
- Memorijske operacije
  - Čitanje, upis, RMW
  - Izdate kada napuste okruženje procesora i uđu u memorijski sistem
- Šta bi bilo da nema keš memorija (transparentnost)?
  - Svaki pristup se obavlja u memoriji
  - Serijalizacija memorijskih operacija (svi vide isti poredak)
- Preplitanje pristupa od različitih procesora
  - Za svaki procesor – programski poredak
  - Od različitih procesora – poredak nametnut sinhronizacijom
  - Pretpostavka – izvrše se atomski, kao celina

Koherentni memorijski sistem

- Koherentni memorijski sistem
  - Za svaku lokaciju rezultati izvršenja impliciraju neki ukupni serijski poredak svih operacija za tu lokaciju
  - Operacije jednog procesa se javljaju u programskom poretku u kojem su i izdate
  - Vrednost koju vraća čitanje je vrednost upisana poslednjim upisom u serijskom poretku u tu lokaciju

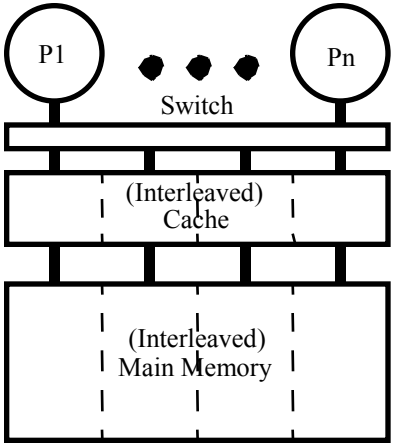


Koherentni memorijski sistem

- Memorijski sistem je koherentan ako
  - Ako čitanje od procesora P za adresu X,
    koje sledi upis od P na istu adresu,
    vraća vrednost upisanu ovim upisom
  - Ako čitanje od nekog procesora za adresu X,
    koje sledi upis od drugog procesora na istu adresu,
    vraća vrednost upisanu ovim upisom
    (... ako su dovoljno vremenski udaljeni)
  - Efekti dva upisa od različitih procesora
    vide se u istom poretku od strane svih procesora
- Uslovi i osobine koje implicira
  - Programski poredak
  - Propagacija upisa čini ga vidljivim drugim
    (... ali se ne kaže kada!)
  - Serijalizacija upisa – svi vide isti poredak!

Zajednička keš memorija

- Prednosti
  - Izbegava problem koherencije
  - Štedi ukupni prostor (jedna kopija umesto više njih)
  - Pozitivna interferencija (jedan prihvata, više njih koriste)
  - Niska cena komunikacije (2-10 ciklusa)
  - Mogući veći blokovi bez "lažne deljivosti"
  - Koristi preklapanje radnih skupova podataka
  - Često se deli L2 ili L3 keš memorija
- Nedostaci
  - Ograničenje propusnog opsega
  - Povećana latencija za sve pristupe
    - Krosbar + veća keš memorija
    - Utiče na takt procesora
  - Negativna interferencija (jedan izbacuje podatke potrebne drugom)
  - Složeniji dizajn



Rešavanje problema koherencije

- Protokoli za koherenciju keš memorija
  - Kompletan skup operacija za pristup zajedničkoj memoriji koji obezbeđuje njen konzistentan izgled
  - Skup mogućih stanja u keš memorijama
    + stanje u memoriji
    + prelazi između stanja
- Veliki spektar rešenja
- Zahtevi za efikasnošću
  - Minimizacija vremena pristupa
  - Minimizacija "*overhead*"-a za održanje koherencije

Klasifikacija rešenja

- Priroda rešenja
  - Hardverska
  - Softverska
- Odgovornost za održavanje koherencije
  - Lokalna
  - Globalna
- Organizacija informacije vezane za koherenciju
  - Centralizovana
  - Distribuirana
- Strategija obaveštavanja
  - *Broadcast*
  - *Multicast*
  - *Unicast*

Klasifikacija rešenja

- Strategija održavanja koherencije
  - Na upis u kopiju deljenog podatka
    - Poništavanje (*invalidate*)
    - Ažuriranje (*update*)
  - Na promašaj pri čitanju deljenog podatka, on se dobija iz
    - Operativne memorije (*memory supply*)
    - Keš memorije (*cache supply*)
- Granularnost objekta koherencije
  - Logička (segment)
  - Fizička
    - Reč
    - Blok (linija)
    - Stranica

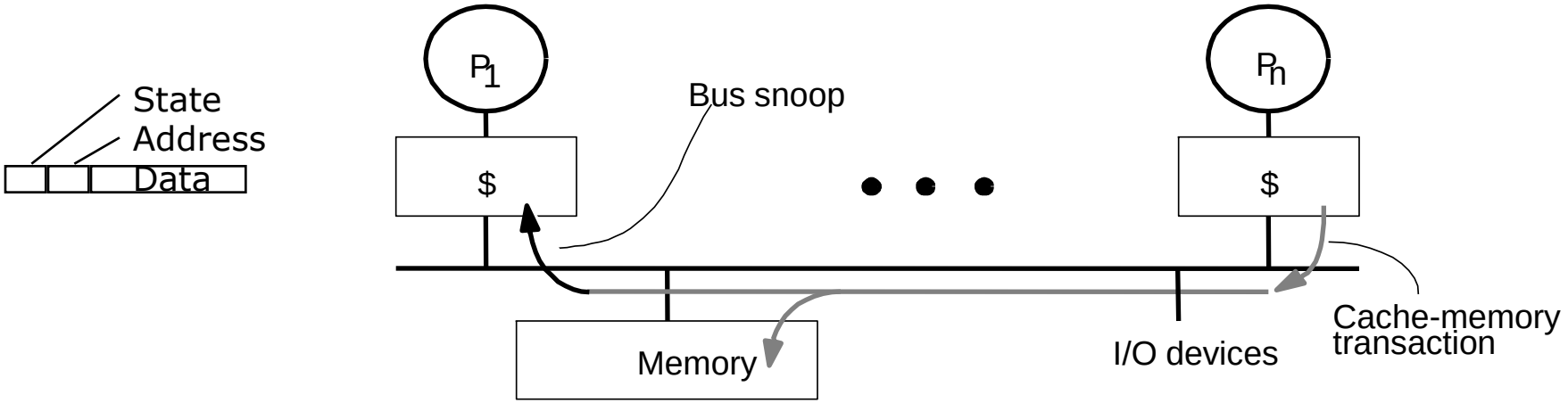
Softverska rešenja

- Zasnivaju se na akcijama
  - ... programera, prevodioca, OS, *run-time* sistema
  - Mehanizmi: označavanje stranica/blokova koje ne idu u keš, *flush* i *invalidate* instrukcije, itd.
- Prednosti
  - Skalabilnost
  - Nezavisnost od ICN
  - Manja cena?
- Nedostaci
  - Forsiraju deljivost kroz memoriju
  - Statička rešenja često konzervativna
  - Zahtevaju napredna softverska sredstva
    ... ali, ponekad, i značajnu hardversku podršku!

Hardverska rešenja

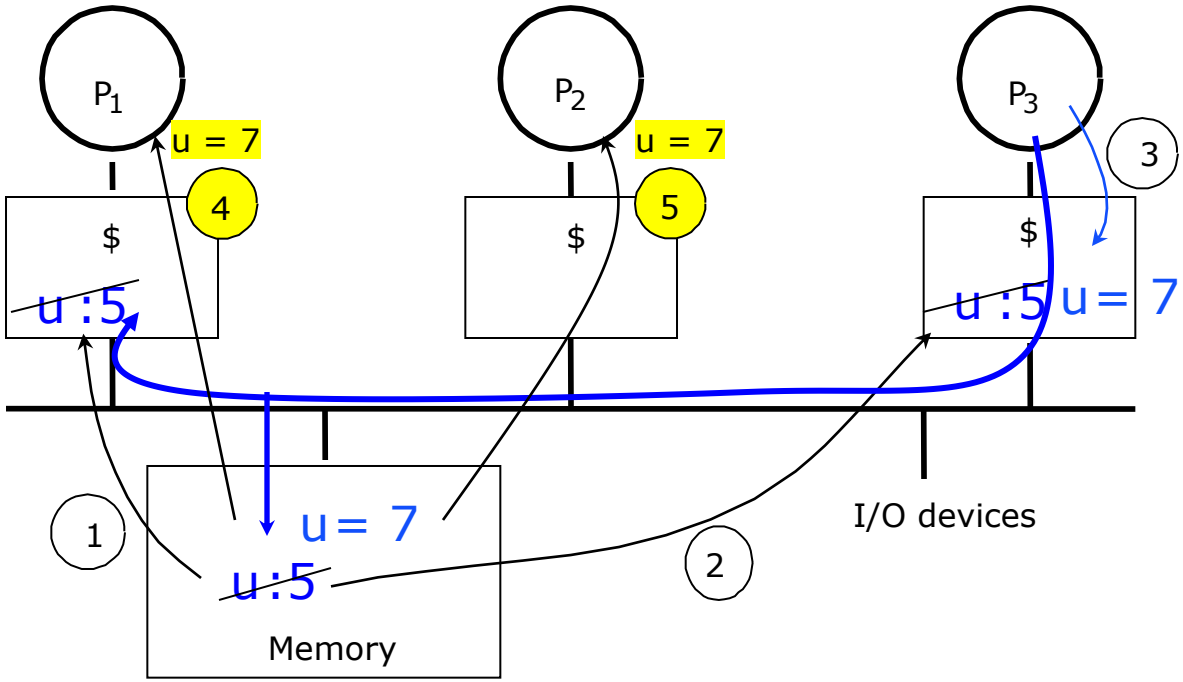
- Potpuno zasnovana na HW mehanizmima
- Prednosti
  - Transparentnost za korisnika i SW
  - Dinamičke odluke u održavanju koherencije
  - Potencijalno bolje performanse
  - Ne ograničavaju migraciju procesa
  - Dobro prilagođena arhitekturi sistema
- Nedostaci
  - Povećana HW složenost
- Podela protokola
  - Distribuirani (*snoopy*)
  - Centralizovani (*directory*)

Snoopy protokoli



- Prilagođeni sistemima sa zajedničkom magistralom
- Karakteristike
  - Informacija bitna za koherenciju potpuno distribuirana
  - Odgovornost na lokalnim kontrolerima keš memorija
  - Obaveštavanje o bitnim akcijama "*broadcast*"-om
  - Nadgledanje magistrale (*snooping*)

Primer: WTI (*Write-Through Invalidate*)



Osnovne projektne odluke

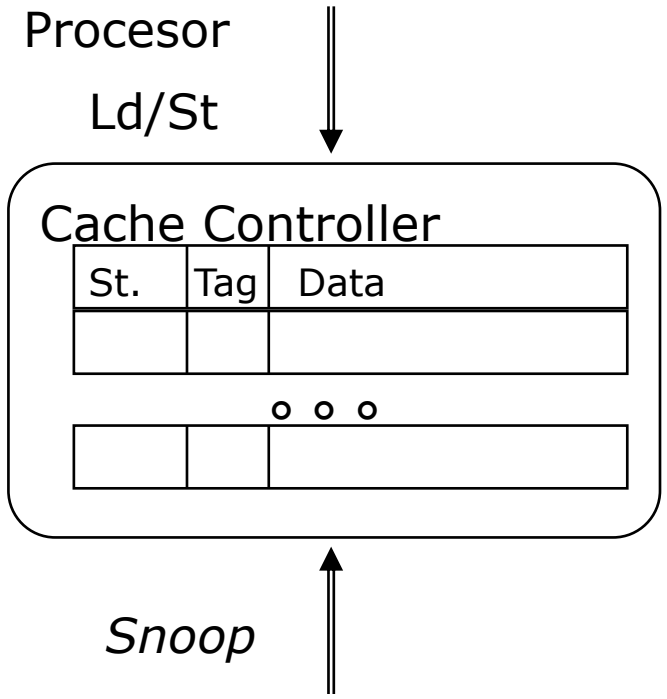
- Transakcije na magistrali
  - Osnovna apstrakcija projektovanja sistema
  - Protokol: arbitracija, komanda/adresa, podaci
  - Magistrala je i sredstvo serijalizacije i sinhronizacije

  => Svaki uređaj vidi svaku transakciju

- Dijagram stanja prelaza
  - Stanja (raspoloživost bloka i mogućnosti pristupa)
    - Npr. WTI (invalid, valid)
    - Npr. WBI (invalid, valid, dirty)
  - Prelazi između stanja - reakcije na akcije:
    - Procesora
    - Transakcije sa magistrale
  - Automat stanja (FSM)

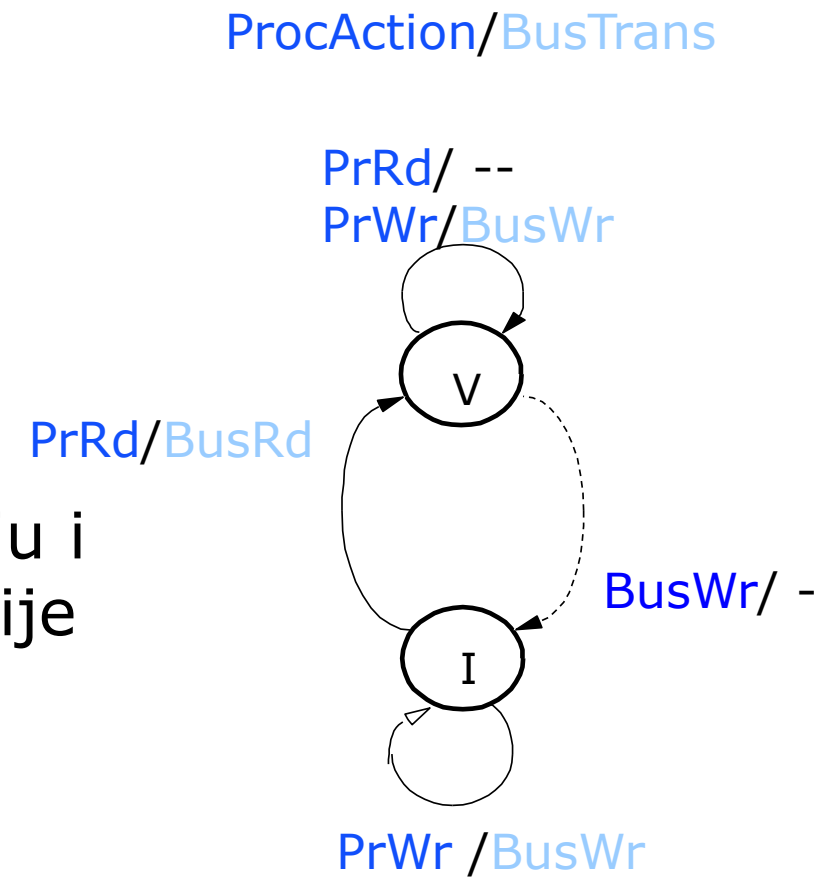
Osnovne projektne odluke

- "*Snoopy*" protokol
  - Skup stanja
  - Dijagram prelaza stanja
  - Akcije
- Implementacija
  - Kontroler
  - Magistrala
  - Često dupli tagovi
- Osnovne odluke
  - Trenutni (*write-through*) ili odloženi (*write-back*) upis
  - Poništavanje ili ažuriranje



WTI Protokol

- Dva stanja za svaki blok
  - 1 bit - Valid(V) i Invalid(I)
  - Kao u jednoprocesorskim
  - Globalno stanje bloka je unija lokalnih stanja
  - Blok koji nije prisutan tretira se kao nevažeći
- Svaki upis ide u memoriju i poništava sve ostale kopije
  - Strategija MRSW
  - "*write-no-allocate*"
- Vrlo jednostavan



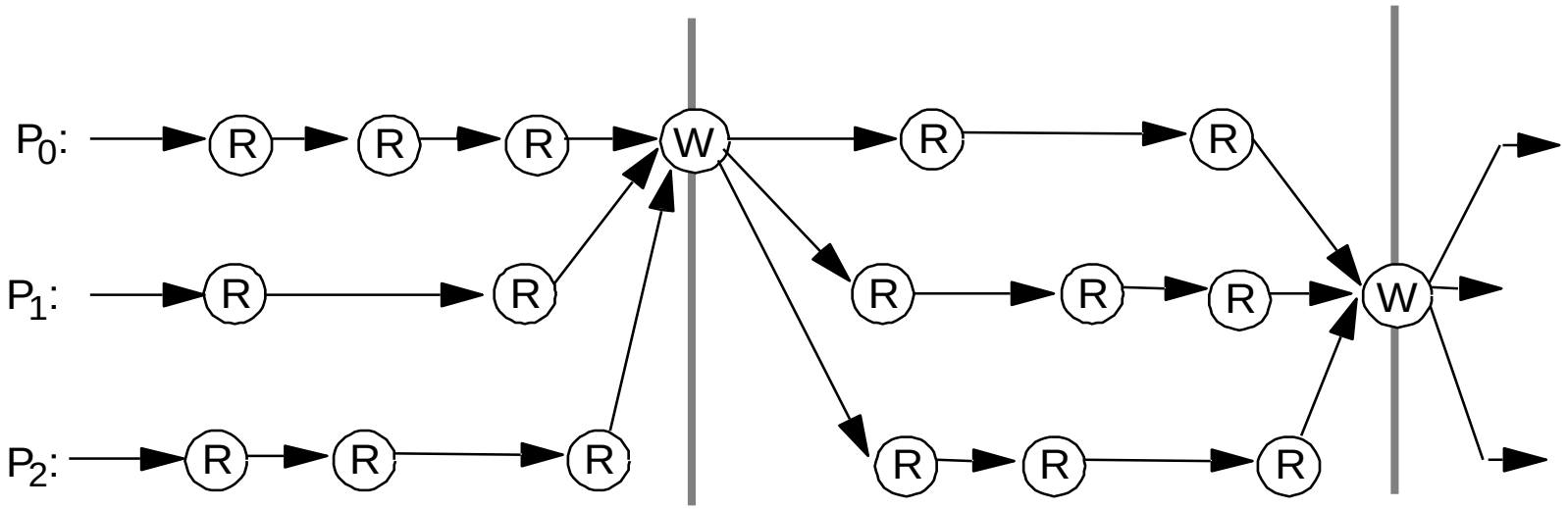
WTI protokol: dokaz koherencije

- Pretpostavka – atomske operacije + L1 keš
  - Transakcija na magistrali neprekidna
  - Procesor čeka završetak prethodne memorijske operacije pre nego što izda sledeću
  - Invalidacija se izvrši tokom transakcije na magistrali
- Svi upisi izlaze na magistralu
  - => serijalizacija upisa i invalidacija u poretku magistrale
- Čitanja vide efekat upisa
  - Čitanja od svih procesora moraju videti isti serijalizovani poredak upisa
  - Nezavisna i konkurentna
  - Ne izlazi svako čitanje na magistralu (RH)
  - Nisu potpuno serijalizovana, kao se pojavljuju u poretku?

WTI protokol: dokaz koherencije

- Promašaj pri čitanju (RM)
  - Izlazi na magistralu
  - Vidi efekat poslednjeg upisa u poretku
- Pogodak pri čitanju (RH)
  - Ne izlazi na magistralu
  - Dobija vrednost
    - Poslednjeg upisa od istog procesora
    - Poslednjeg RM u programskom poretku
  - Obe ove transakcije su se pojavljivale na magistrali
  - RH vidi vrednosti proizvedene u istom poretku magistrale
- Programski poredak i poredak magistrale obezbeđuju koherenciju

WTI protokol: dokaz koherencije



- Ne ograničava poredak čitanja različitih procesora između upisa