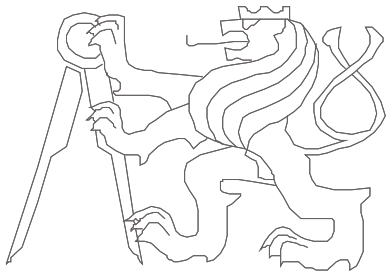
Pokročilé architektury počítačů

Multiprocesorové systémy

SMP a problém koherence

V přednášce jsou použity obrázky z knihy D.E.Culler, J.P. Singh, A.Gupta: Parallel Computer Architecture: A HW/SW Approach, Morgan Kaufmann Publishers, 1998



České vysoké učení technické, Fakulta elektrotechnická

Osnova

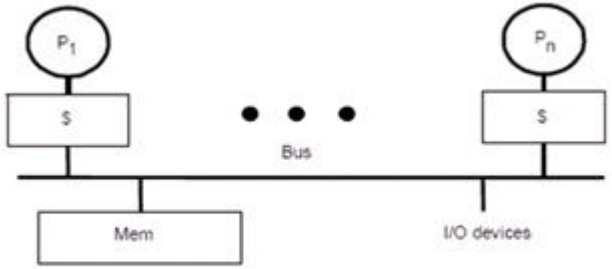
- Co je to skrytá paměť?
- Co to jsou SMP?
- Jsou i jiné MP systémy? UMA, NUMA, aj.
- Konzistence, koherence
- Koherenční protokoly
- Možné stavy dat ve skryté paměti

Multiprocesorové systémy

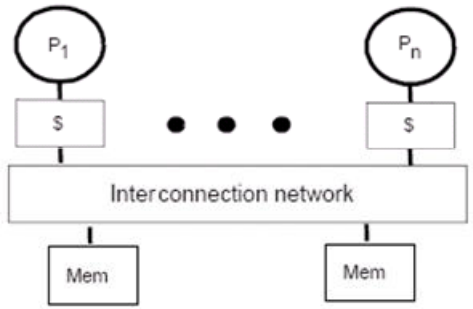
- **Softwarový** pohled (připomeňte si přednášku č.2)
  - Procesy se sdílenou pamětí (shared memory system - SMS). Na počítači běží jedna instance OS (společné plánování), Standard OpenMP, taktéž MPI.
    - Výhody: snazší programování, nižší latence, HW podpora konzistence skryté paměti.
  - Procesy s oddělenou pamětí (DMS): komunikují pomocí předávání zpráv – message passing. Na každém uzlu (procesoru, skupině procesorů - hybridní) jiná instance OS. Síťové protokoly, RPC, Standard MPI.
    - Výhody: méně potřebného HW, snazší škálování.

Multiprocesorové systémy

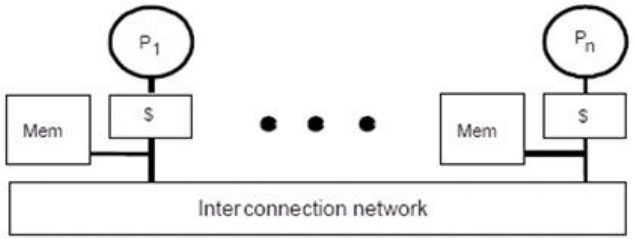
- Multiprocesorové systémy, systémy s více procesory. Ale mohou to být i procesory „jen” s více jádry.
- **Hardwarový** pohled:
  - Systémy se sdílenou fyzickou pamětí (společný fyzický adresový prostor) – SMP,
  - Systémy s oddělenou fyzickou pamětí (každý uzel má nezávislý fyzický adresový prostor) – UMA, NUMA, ale i Clustery, NOW, metapočítače.



tradiční SMP

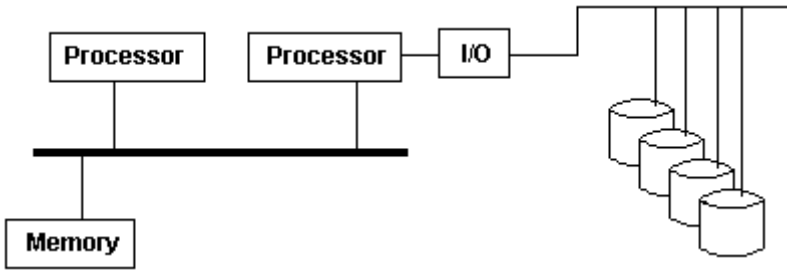


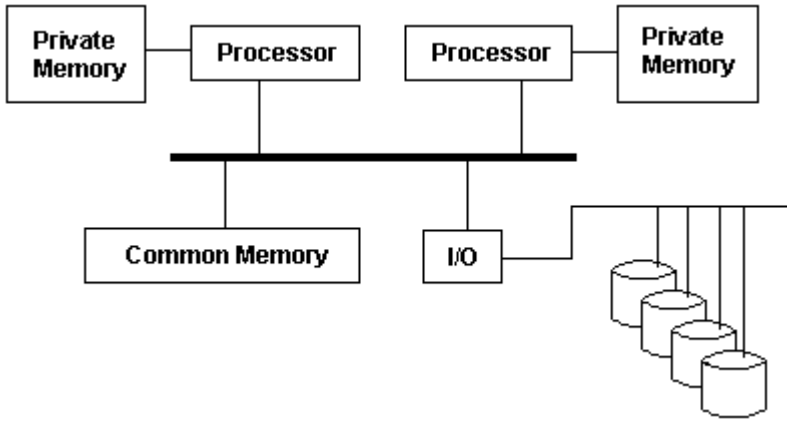
UMA

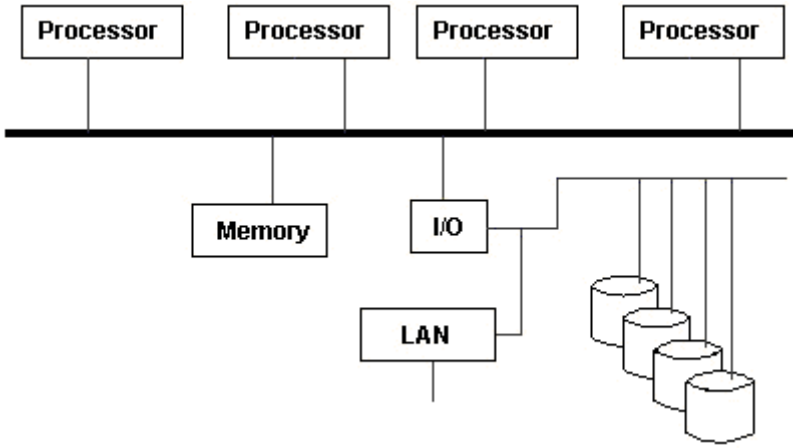


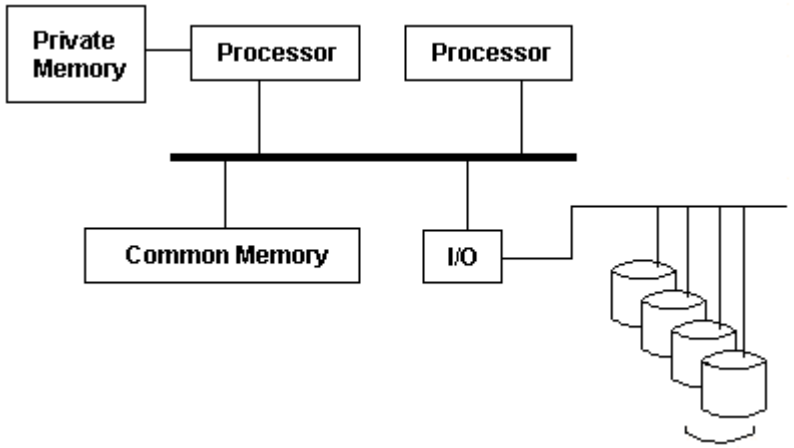
NUMA

Najdete SMP ???


Processor
Processor
I/O
Memory


Private Memory
Processor
Processor
Private Memory
Common Memory
I/O

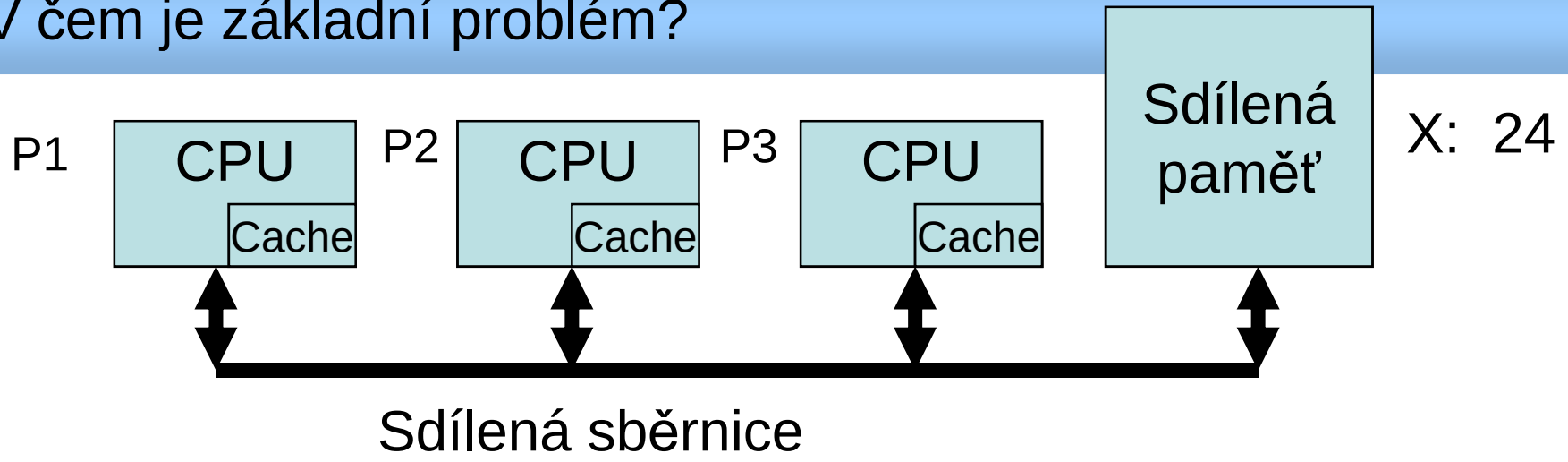

Processor
Processor
Processor
Processor
Memory
I/O
LAN


Private Memory
Processor
Processor
Common Memory
I/O

Systémy se sdílenou fyzickou pamětí

- Hlavní téma dnešní přednášky.
- Přirozené rozšíření jednoprocesorových počítačů přidáním dalších procesorů (či jader).
- Tradiční jednoprocesorové OS jsou modifikovatelné. Použijí se k plánování procesů pro více procesorů.
- Víceprocesový/vícevláknový program poběží na systémech s jedním procesem v režimu sdílení času (Timesharing). Využije ale i více procesorů, pokud jsou k dispozici.
- Vhodný model pro úlohy s převažujícím sdílením dat, ta se sdílejí automaticky. Toto řeší transparentně HW.
- Obtížně se ale škáluje (pro větší počty procesorů).

V čem je základní problém?



- Procesor P1 čte X (=24) a zapíše do své skryté paměti.
- Procesor P2 čte X (=24) a zapíše do své skryté paměti.
- Procesor P1 zapíše X :=31. Provede se modifikace jeho SP.
- Jakou hodnotu teď přečte P3 z X?
  - X=24. Totéž přečte P2.
  - P1 ale přečte X=31!
- **Stačí** na odstranění problému **Write-throuhg? Ne!**

Co jsme zjistili?

- Problém spočívá v paměťové nekoherenci:
- Procesor data (správně) modifikoval ve své skryté paměti.
- Změna obsahu hlavní paměti nestačí.
- Skryté paměti ostatních procesorů mohou obsahovat neaktuální data.

Důležité pojmy

- Paměťová koherence
  - Pravidla pro přístupy k jednotlivým paměťovým místům.
- Paměťová konzistence
  - Pravidla pro všechny paměťové operace.

Paměťová koherence

- Definice: Řekneme že multiprocesorový paměťový systém je **koherentní** jestliže
  - výsledek jakéhokoli provádění programu je takový, že pro každé paměťové místo je možné sestavit myšlené sériové pořadí čtení a zápisů k tomuto paměťovému místu a platí:
  - 1. Paměťové operace k danému paměťovému místu pro každý proces se provedou v pořadí, ve kterém byly tímto procesem spuštěny .
  - 2. Hodnoty vrácené každou operací čtení jsou hodnotami naposledy provedené operace zápis do daného paměťového místa s ohledem na sériové pořadí.
- Metody pro zajištění koherence nazýváme koherenční protokoly.

Paměťová konzistence

- Existuje více konzistenčních modelů, nejznámější je model **sekvenční konzistence** (SC).
- Definice (Lamport, 1979): “Počítač je sekvenčně konzistentní, jestliže je výsledek provádění programu stejný, jako kdyby operace na všech procesorech byly provedeny v nějakém sekvenčním pořadí a operace každého jednotlivého procesoru se objevují v této posloupnosti v pořadí daném jejich programem.”
- Sekvenční konzistence je důležitým teoretickým modelem pro dokazování korektnosti paralelních algoritmů.

Postačující podmínky pro zajištění SC

- Každý procesor P(i) spouští paměťové operace v programovém pořadí.
- Procesor P(i), který spustí operaci Write, nespustí další paměťovou operaci dříve, než se tato dokončí.
- Procesor P(i), který spustí operaci Read, nespustí další paměťovou operaci dříve, než se tato dokončí a než se dokončí operace Write, jejíž hodnotu vrací operace Read.

Zajištění koherence – koherenční protokoly

- Přímý zápis – **Write-through** je
  - současný zápis slova do paměťového bloku skryté paměti i na odpovídající místo v hlavní paměti.
- Jednoduchý, ale pomalý způsob udržování shodného obsahu SP a paměti, problém však neřeší..
- Zatěžuje komunikaci s pamětí, vyžaduje zapisovací vyrovnávací paměť – Write Buffer.
- Protokol **Write-through** je pro rozsáhlejší systémy prakticky nepoužitelný, není škálovatelný.
- reálně používané protokoly jsou MESI, MSI, MOESI, MESIF.

Připomenutí

- **Cache** – vyrovnávací, nebo lépe **skrytá paměť**. Je umístěna mezi procesorem a hlavní pamětí. Zmírňuje disproporci mezi rychlostí procesoru (vysokou) a paměti (nízkou).
- **Write-through** protokol (pomalejší) k zajištění shody obsahu skryté a hlavní paměti. Spočívá v současném zahájení zápisu dat do skryté i hlavní paměti.
- **Write-back** je jiný protokol k zajištění téhož. Do hlavní paměti se zapisuje až tehdy, kdy by se o blok ze SP mohlo přijít. Je z principu rychlejší.

Řešení problému koherence paměti v SMP?

- Snooping
  - Snooping = slídění, špehování.
  - Vyžaduje doplnění SP o HW, který sleduje dění na sběrnici a
    - Detekuje relevantní transakce,
    - Aktualizuje stavy relevantních bloků dat.

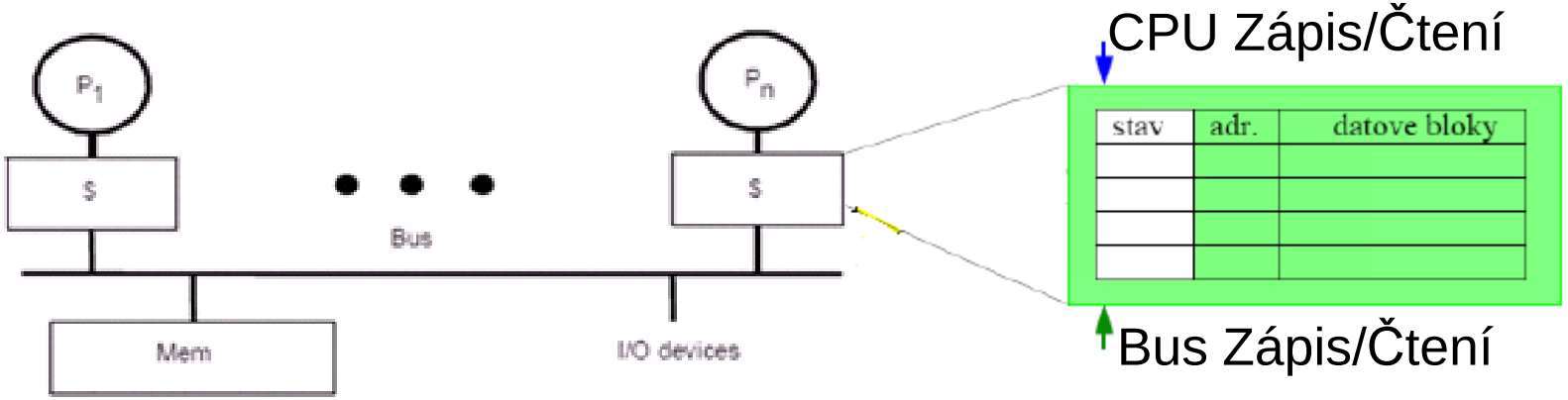
- Directory based
  - když není k dispozici sdílená sběrnice (nebo ring)

Bus Snooping

- Varianta, kdy každý procesor ví, kdo má kopii jeho ve SP uložených dat, je příliš složitá. Proto

- Řadič každé SP spojitě slídí (snoops) na sběrnici po zápisech týkajících se dat na adresách, které obsahuje.
- To vyžaduje, aby byla struktura sběrnice globální, aby ji viděli všichni.
- Právě zde je problém ve škálovatelnosti.
- Lépe škálovatelné řešení je právě „Directory based“.

Doplněná skrytá paměť



Možné stavy

Stav bloku	Hodnota platná	Existuje další kopie v jiné cache	Hlavní paměť má akt. hodnotu
Invalid	Ne	*Možná*	*Možná*
Exclusive	Ano	Ne	Ano
Shared	Ano	Ano (*možná*)	Ano
Modified	Ano	Ne	Ne
Owned	*Ano*	Ano (*možná*)	*Ne*

Použito v MOESI protokolu

Snooping protokoly

- Zápis se zneplatněním
  - Procesor zapisuje na adresu. To musí vyvolat zprávu o provedení zneplatnění obsahu na všech místech, které stejnou kopii obsahují.
  - Všechny slídící řadiče SP provedou zneplatnění ve svém obsahu.
  - Provede se i změna v paměti. Předpoklad: nechť je náš SP protokol Write-through.
  - Všechna možná čtení v ostatních procesorech teď narazí na výpadek ve SP a musí data načíst z paměti.

Snooping protokoly

- Zápis s aktualizací
  - Procesor, který zapisuje, musí nejprve sběrnici získat. Pak na ní vyšle nová data.
  - Všechny úspěšně slídící řadiče SP aktualizují svůj obsah.
- V obou popsaných strategiích může nastat problém: sběrnici může v arbitraci získat jen jeden procesor. Třeba i jiný….
- Proto tyto protokoly nestačí. Je zapotřebí jiný…

Základní Snoopy Protokol, shrnutí faktů

- Se zneplatněním, Write-back skryté paměti.
- Každý paměťový blok je v jednom ze stavů:
  - Je ve SP (alespoň jedné), je aktualizován v paměti (Shared)
  - Nebo je Dirty právě v jedné SP (Exclusive)
  - Nebo v žádné SP není.
- Každý blok SP je právě v jednom stavu:
  - Shared : blok se může číst
  - nebo Exclusive : SP je v jediné kopii, je zapisovatelná a umazaná (dirty)
  - Nebo Invalid : neobsahuje data
- Výpadek čtení (Read miss) způsobí, že všechny SP začnou slídit (snoop) na sběrnici.
- Zápisy související s čištěním linky se zpracují jako výpadky.

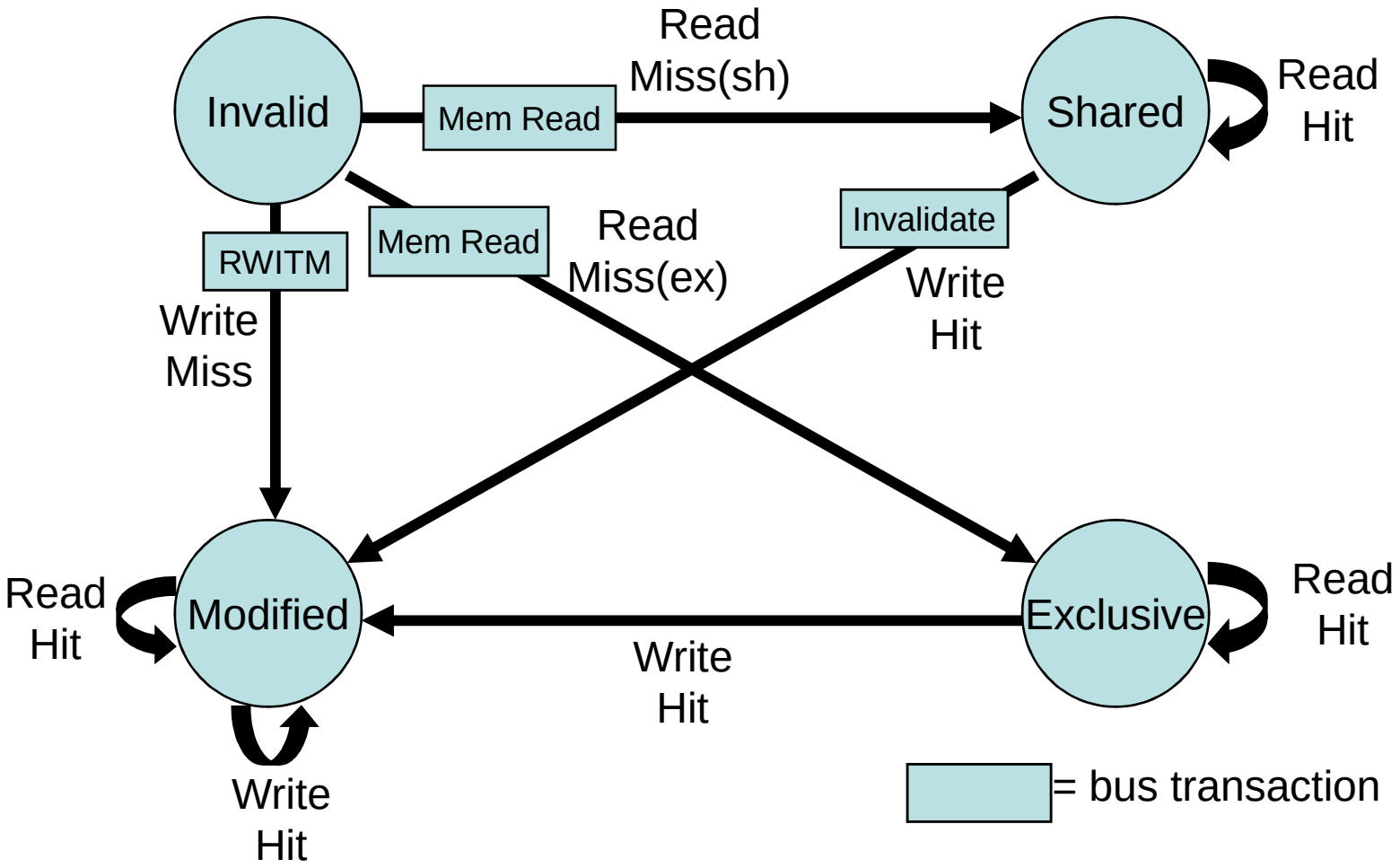
Zneplatnění či aktualizace: co je lepší?

- Nový protokol smí sběrnici zatěžovat co nejméně. Je takový?
- MESI
- Kdo jej vymyslel?
- Znám je pod označením *Illinois protocol,* protože odtud, University of Illinois at Urbana-Champaign, pochází.

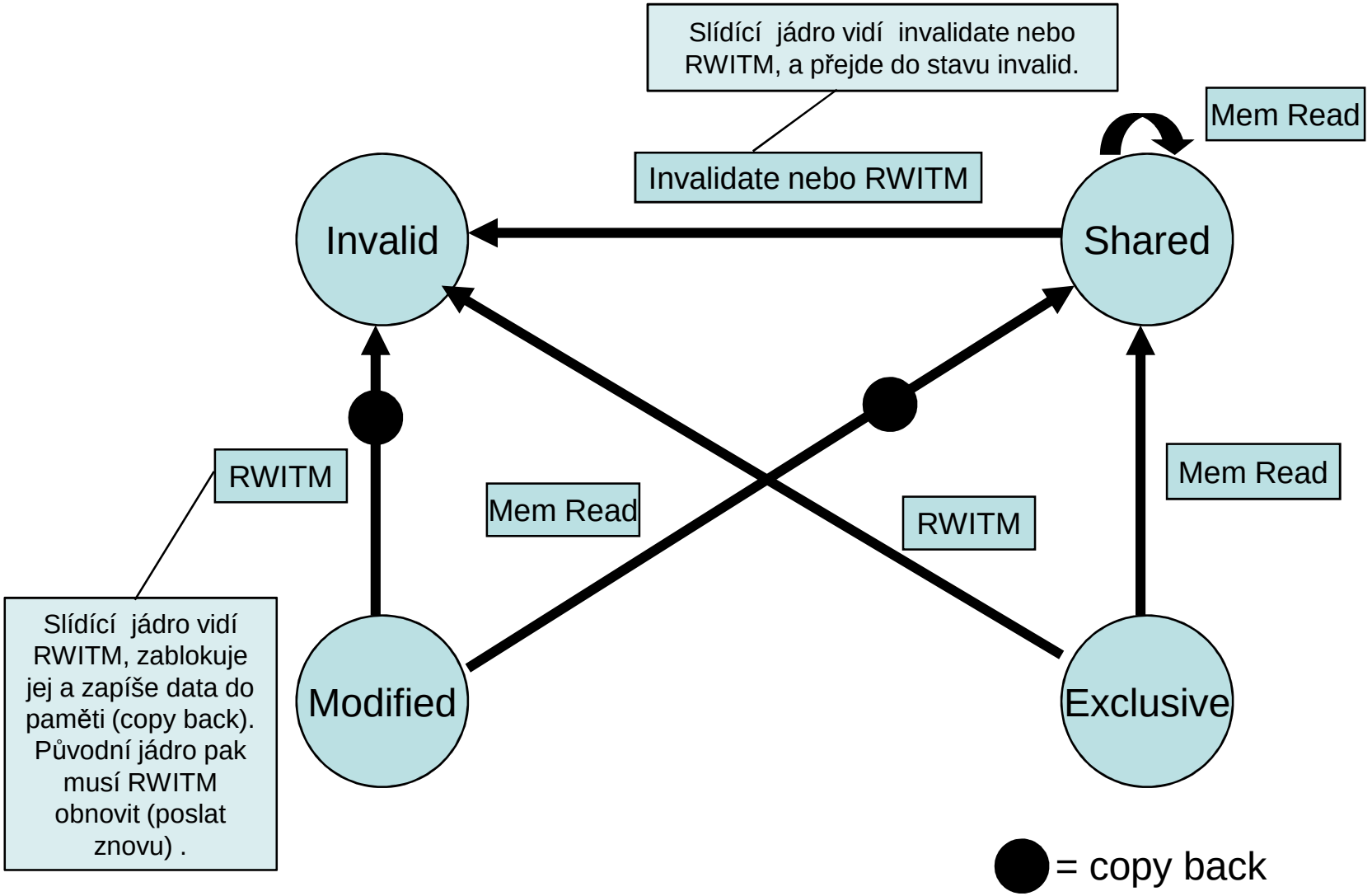
MESI

- Kompaktně popsané akce shrnuje stavový diagram přechodů. Ten zobrazuje, co se stane s řádkou v cache daného procesoru v případě
  - Přístupu tohoto procesoru k paměti (read hit/miss, write hit/miss)
  - Přístupu k paměti jiným procesorem, který úspěšně vyslídí tento řadič cache (Mem read, RWITM = Read With Intent To Modify, Invalidate).
- Akce připojeného/lokálního procesoru:
  - Read Hit (čtená hodnota je k dispozici)
  - Read Miss (výpadek při čtení)
  - Write Hit (zápis byl úspěšný)
  - Write Miss (výpadek při zápisu)

MESI – locally initiated accesses



MESI – remotely initiated accesses



Poznámky ulehčující implementaci

- Znalost, že ve SP uložená hodnota je E (Exclusive), vede v obou dříve zmíněných schématech na možnost nevysílat žádnou zprávu.
- Zneplatnění předpokládá, že při zápisu hodnoty do SP byla tato zároveň propagována (Writte-through) do paměti. U Copy-back, pak musí jiné procesory obnovit hodnotu při výpadku ve SP.
- Změny stavu řádku ve SP nastávají při události Zápis/Čtení (při přístupu do paměti).
- Událost způsobí buď
  - Aktivita připojeného procesoru (přístup do SP), nebo
  - Jako výsledek úspěšného slídění na sběrnici.
- Změny stavu řádku nastávají jen v případě shody adres.

MESI Protokol (1)

- Jde o protokol, který při zneplatnění hodnoty v multiprocesorovém systému minimalizuje akce sběrnice.
- Je založen na zpětném zápisu, tedy se ve SP neaktualizují špinavé řádky dokud nehrozí, že o blok ve skryté paměti přijdeme.
- Vyžaduje rozšíření příznaků ve SP (tags). Zatím tam byly jen příznaky zneplatnění (Invalid) a špinavý (Dirty).
- **Každý řádek SP může být v jednom z těchto 4 stavů (kóduje se 2 bity)**
  - **M** – Modified. Obsah řádku ve SP se liší od obsahu paměti, (jinak odpovídá běžnému Dirty),
  - **E** – Exclusive. Je jedině v této SP a je stejný, jako v paměti.
  - **S** – Shared. Je stejný, jako v paměti, ale nemusí být jen v této SP.
  - **I** – Invalid. Obsah řádku neplatí.

MESI Local Read Hit – **procesor úspěšně čte**

- Řádek SP musí být v jednom ze stavů M, E či S.

- Čtená lokální hodnota je správná. Byl-li stav M, byla předtím lokálně modifikována.
- Výsledek akce – vrátí hodnotu,
- Stav řádku se nemění.

MESI Local Read Miss (1) - **výpadek při čtení procesoru**

- V lokální SP tato adresa není.
- Dále záleží na tom, zda-li
- Není v žádné skryté paměti
  - Procesor spustí požadavek na čtení z paměti,
  - Přečtená hodnota se v lokální SP označí jako E.
- Jedna jiná skrytá paměť má kopii
  - Procesor spustí požadavek na čtení z paměti,
  - Úspěšně slídící SP vydá kopii obsahu na sběrnici,
  - Požadavek na přístup do paměti se zruší,
  - Do lokální SP žádajícího procesoru se zapíše obsah a
  - Příznaky u obou řádků ve SP se nastaví na S.
- Kopie S existují ve více SP - pokračování

MESI Local Read Miss (2) - **výpadek při čtení procesoru**

- Procesor spustí požadavek na čtení z paměti,
- Jedna (libovolná) úspěšně slídící SP vydá kopii na sběrnici,
- Požadavek na přístup do paměti se zruší,
- Do lokální SP žádajícího procesoru se zapíše žádaná hodnota a
- Příznak u řádku v lokální SP se nastaví na S.
- I ostatní kopie zůstanou S.

- Jedna SP má M kopii
  - Procesor spustí požadavek na čtení z paměti,
  - úspěšně slídící SP vydá kopii na sběrnici,
  - Požadavek na přístup do paměti se zruší,
  - Do lokální SP žádajícího procesoru se zapíše žádaná hodnota
  - Příznak u řádku v lokální SP se nastaví na S.
  - **Zdrojová (M) hodnota se zkopíruje do paměti,**
  - Zdrojová hodnota změní příznak M -> S.

MESI Local Write Hit – **úspěšný zápis procesorem**

- Řádek SP je v jednom ze stavů M, E, S.
- M
  - Zapiš hodnotu do lokální cache, *stačí?*
  - Stav neměň.
- E
  - Zapiš hodnotu do lokální cache,
  - Stav E -> M.
- S
  - Procesor vyšle na sběrnici zneplatnění
  - Úspěšně slídící procesory s S kopií změní S->I
  - Lokální SP se aktualizují,
  - Lokální stav se změní S->M.

MESI Local Write Miss (1) – **výpadek při zápisu**

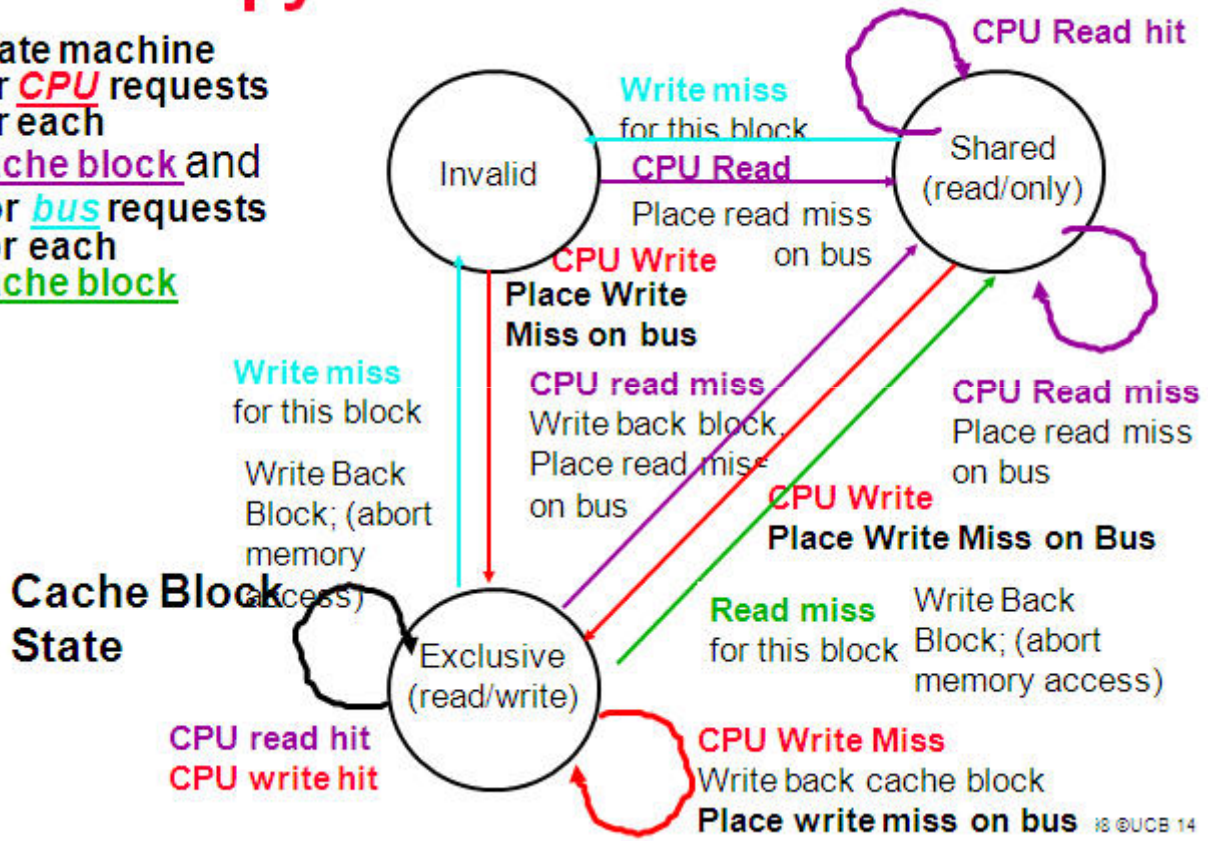
- Skutečná akce závisí na kopiích v jiných procesorech
- Jiné kopie nejsou
  - Načte se řádek do lokální SP,
  - Hodnota se obnoví,
  - Lokální stav příznaku se nastaví na M.
- Jiné kopie jsou. Buď jedna E nebo více S
  - Hodnota se přečte do lokální SP sběrnicovou transakcí RWITM (čtení se záměrem modifikace).
  - Úspěšně slídící procesory to vidí a nastaví I.
  - Lokální hodnota se obnoví a nastaví se příznak M.

MESI Local Write Miss (2) – **výpadek při zápisu**

- Jiná kopie jsou ve stavu M.
  - Procesor vyvolá sběrnicovou transakci RWITM.
  - Úspěšně slídící procesor to vidí a zablokuje tuto transakci,
  - Převezme řízení sběrnice,
  - Zapíše zpět kopii do paměti,
  - Nastaví stav na I.
  - Původní procesor obnoví RWITM požadavek,
  - Tím se přejde na již popsanou situaci Jiné kopie nejsou
  - Načte se řádek do lokální cache,
  - Hodnota se obnoví,
  - Lokální stav příznaku se nastaví na M.

Snoopy-Cache State Machine-III

- State machine for *CPU* requests for each cache block and for *bus* requests for each cache block



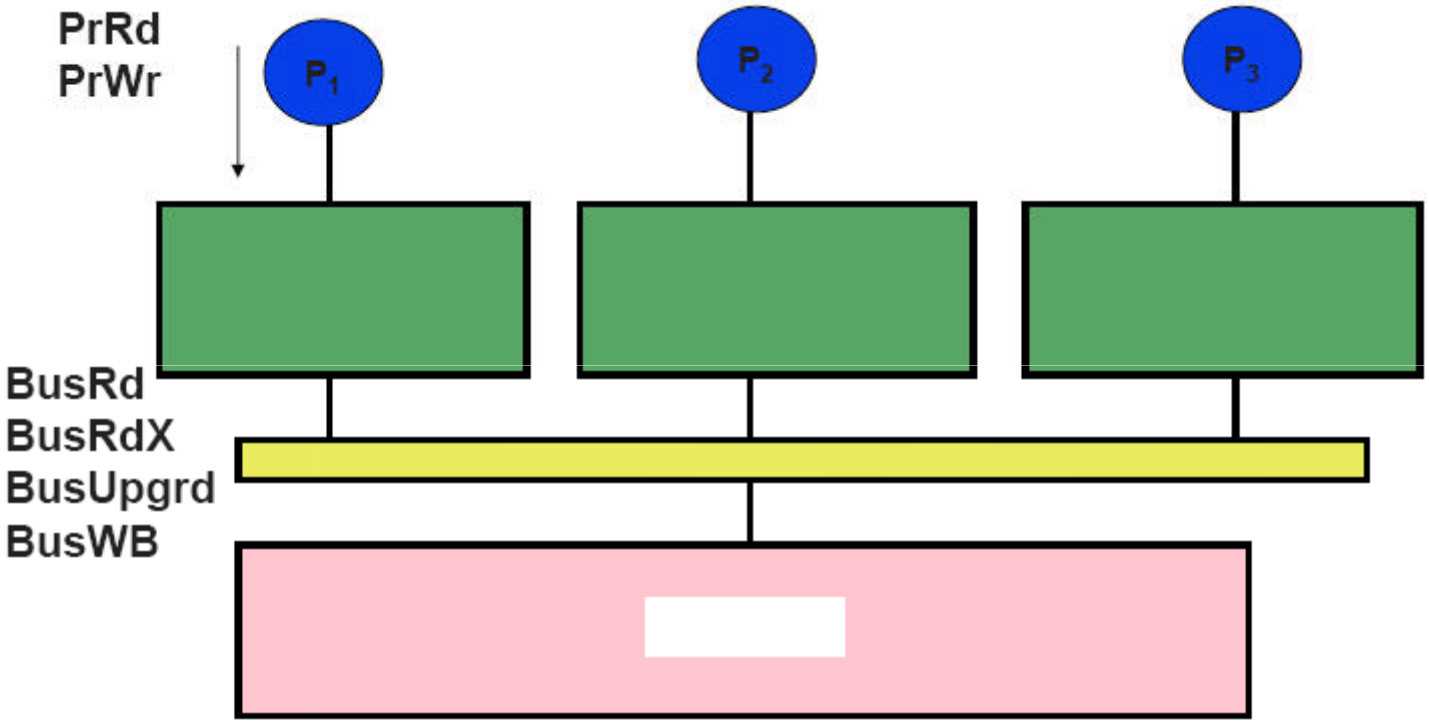
MESI poznámky

- Možné jsou malé odchylky (zvlášť při výpadku při zápisu Write miss).
- Normální zpětný zápis (pokud hrozí, že o blok ve SP přijdeme) se děje, pokud řádek je ve stavu M.
- Vícestupňové SP:
  - Slídit na sběrnici musí jen SP s nejnižší úrovní.

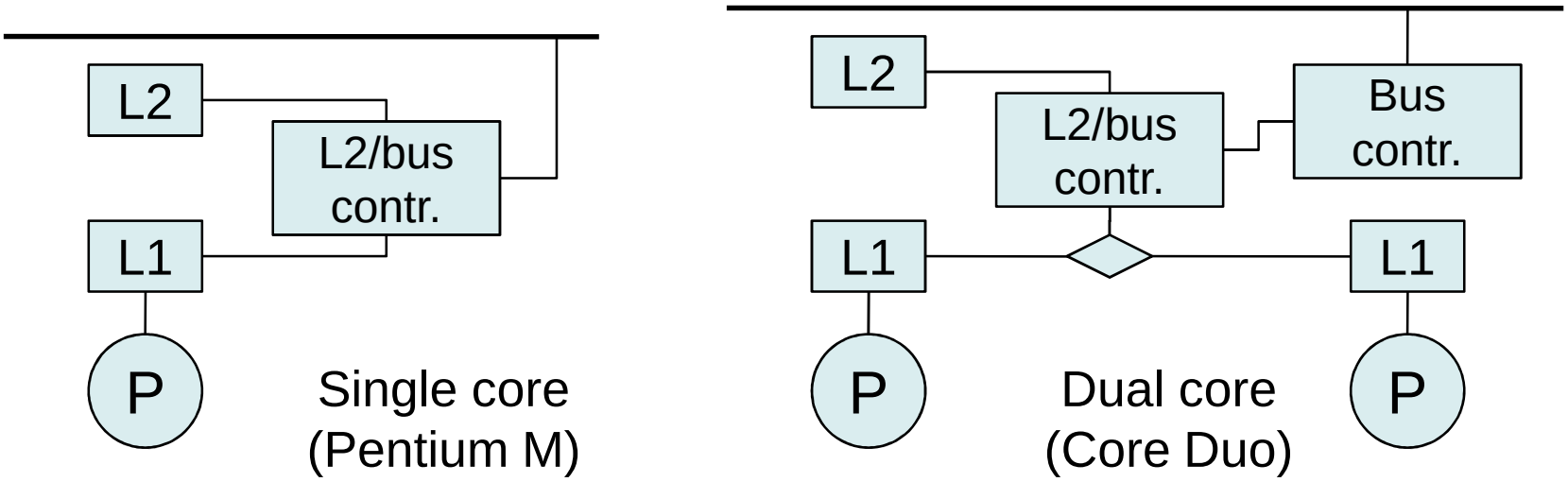
MESI - jiný pohled. Události

- Události - strana procesoru:
  - PrRd – Processor read,
  - PrWr – Processor write.
- Události – transakce na sběrnici:
  - BusRd – čtení bloku do cache (reakce na PrRd na blok ve stavu **Invalid**),
  - BusRdX – čtení se získáním eXclusivity, bude se zapisovat (reakce na PrWr na blok ve stavu **Invalid**)
  - BusUpgrd – upgrade práva na zápis, zneplatnění ostatních kopií (reakce na PrWr na blok ve stavu **Shared**)
  - BusWB – zápis bloku ve stavu **Modified** do paměti při nahrazení jiným blokem. Jde o úklidovou operaci a neovlivňuje koherenci, neboť pouze mění místo, kde je uložena aktuální hodnota daného bloku.

Systém SMP, každý s jednou SP a společnou sběrnicí



CMP

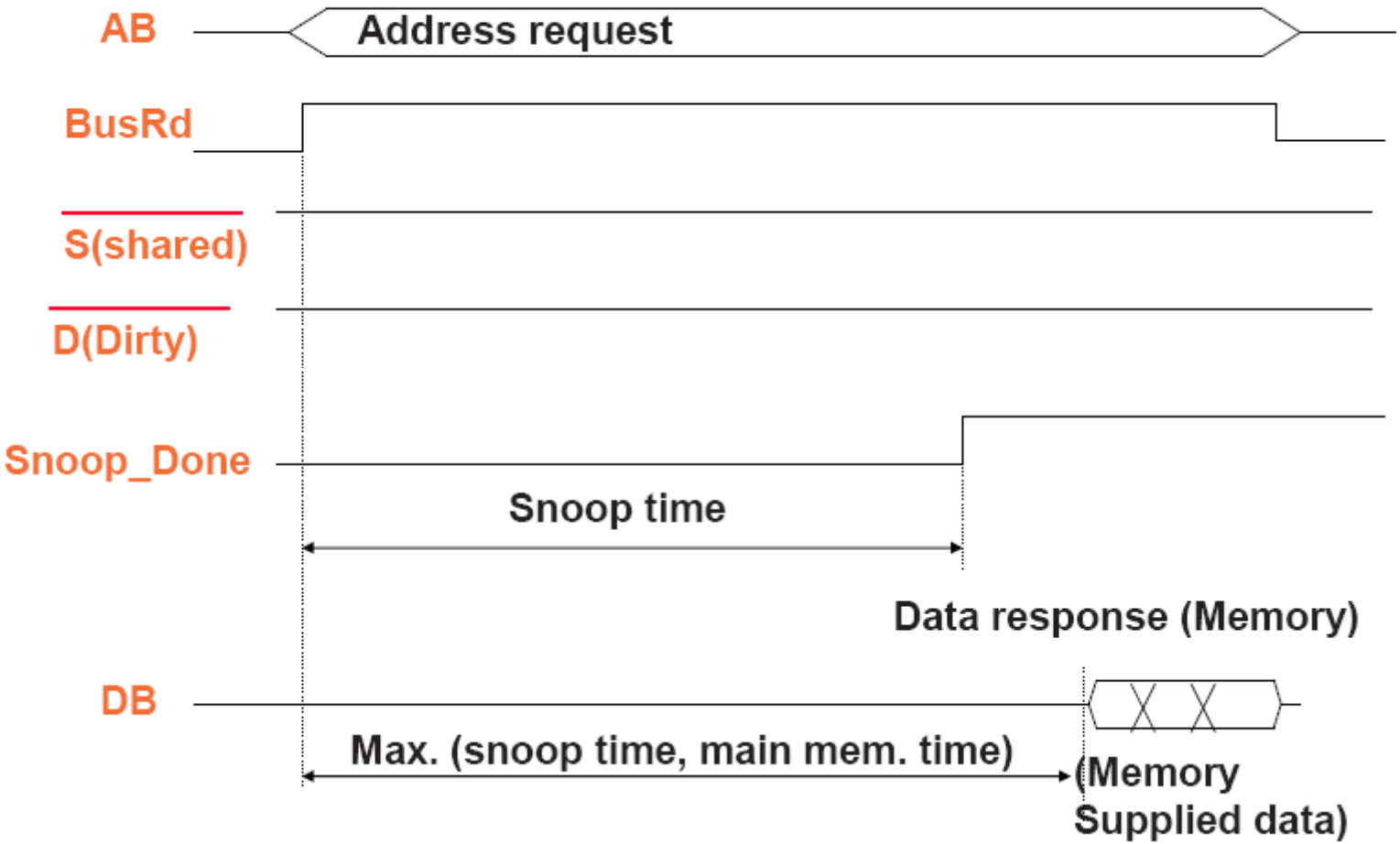


- Pentium M v multiproc. systému na sběrnici – MESI (snooping musí být nad oběma cache – není inkluzivní
- Core Duo:
  - Buss read – nejdřív snoopujeme v L1, v případě missu pak do L2 přes controler
  - Buss write – zneplatnění jdou do L1, L2, a když potřeba i na Bus

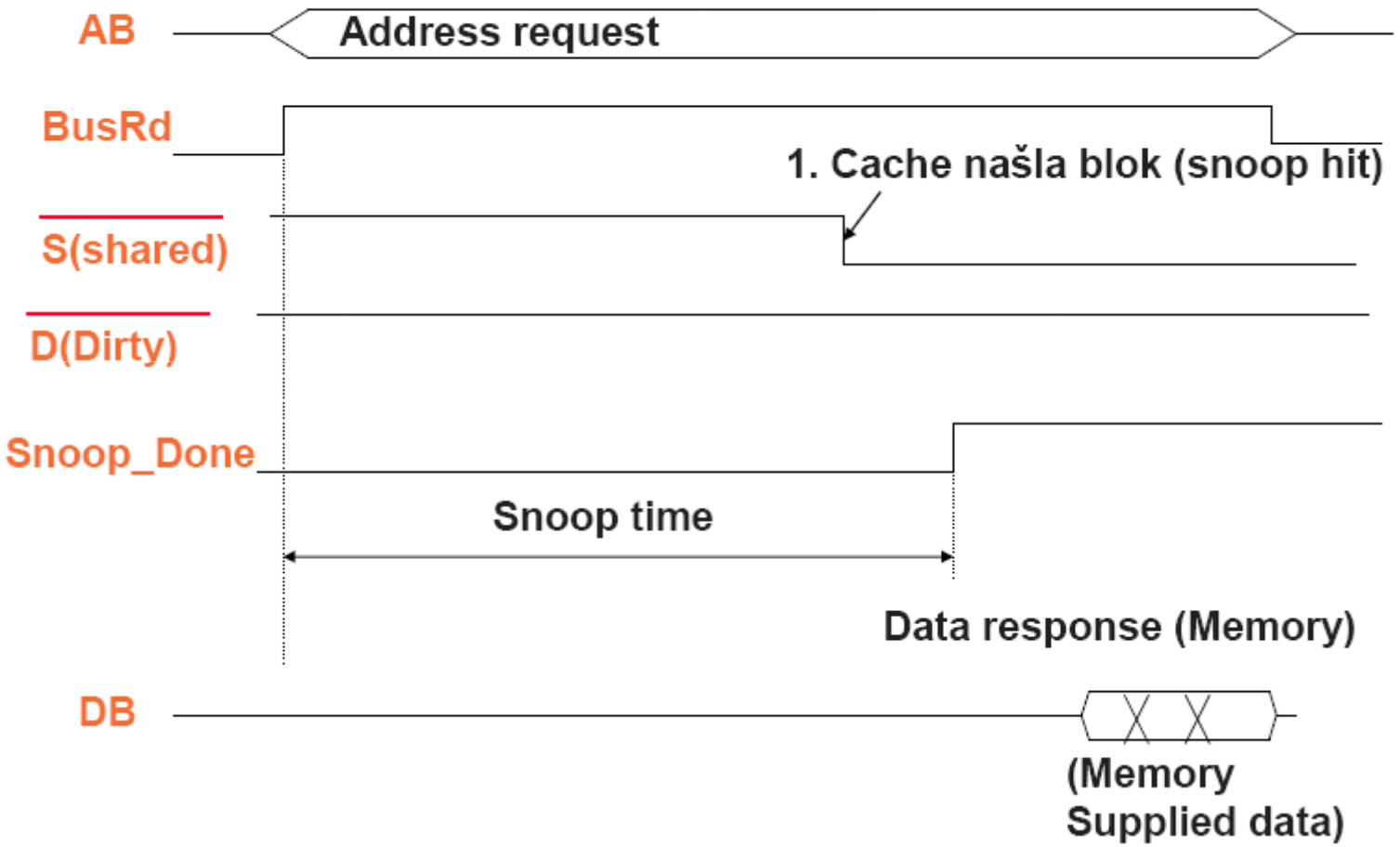
Sdílená sběrnice

- Adresová, datová a řídicí sběrnice.
- Snoop signály (wired OR):
  - S (shared) = 1 pokud některá cache systému má kopii daného bloku ve stavu Exclusive nebo Shared,
  - D (dirty) = 1 pokud některá cache v systému má kopii daného bloku ve stavu Modified (tzv. špinavá kopie = odlišná od hl. paměti),
  - Snoop_Done = 1 pokud všechny cache v systému dokončily snooping (hodnoty na S a D jsou platné) Jde o Wired AND.

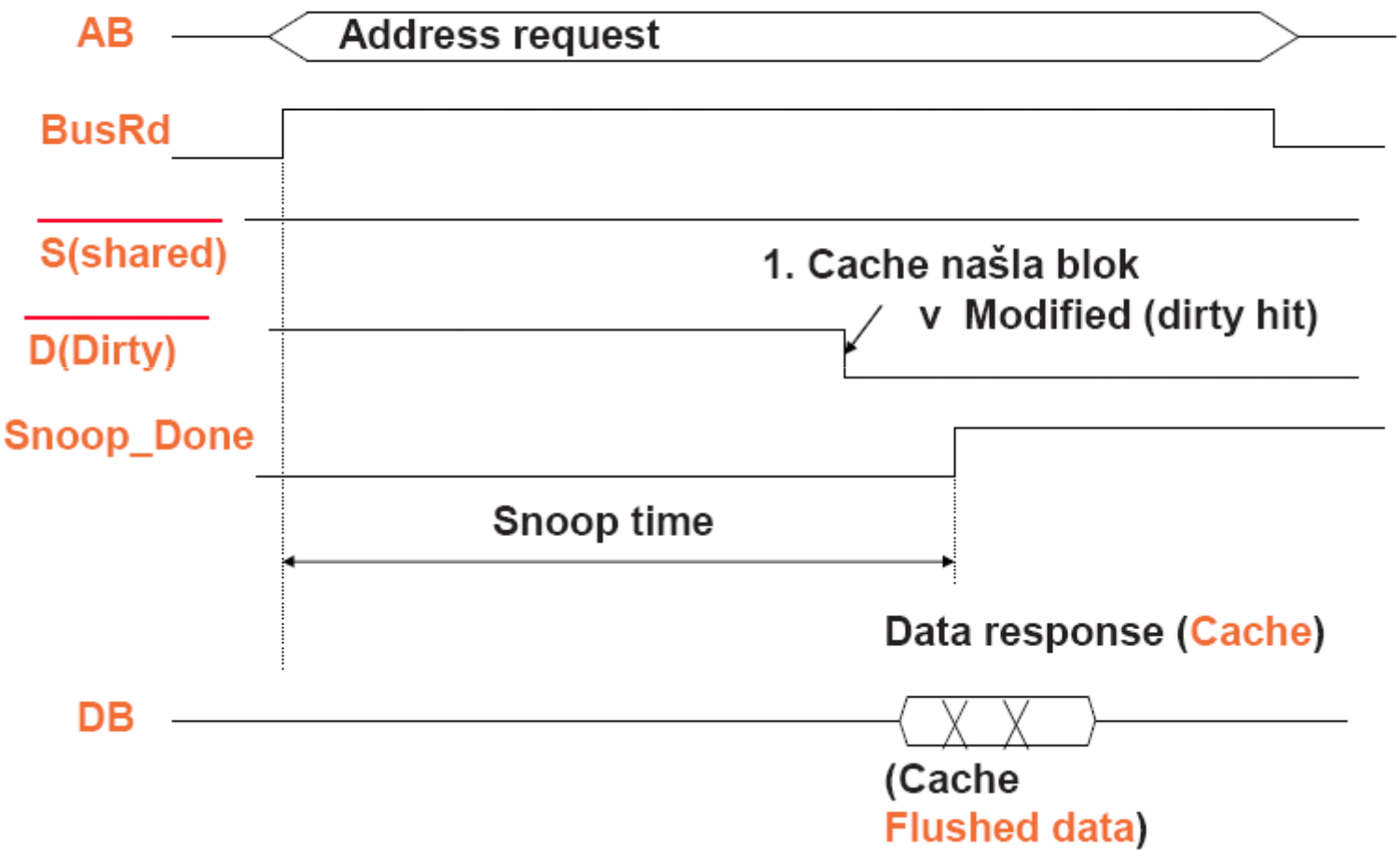
Příklad transakce – BusRd do *Exclusive*



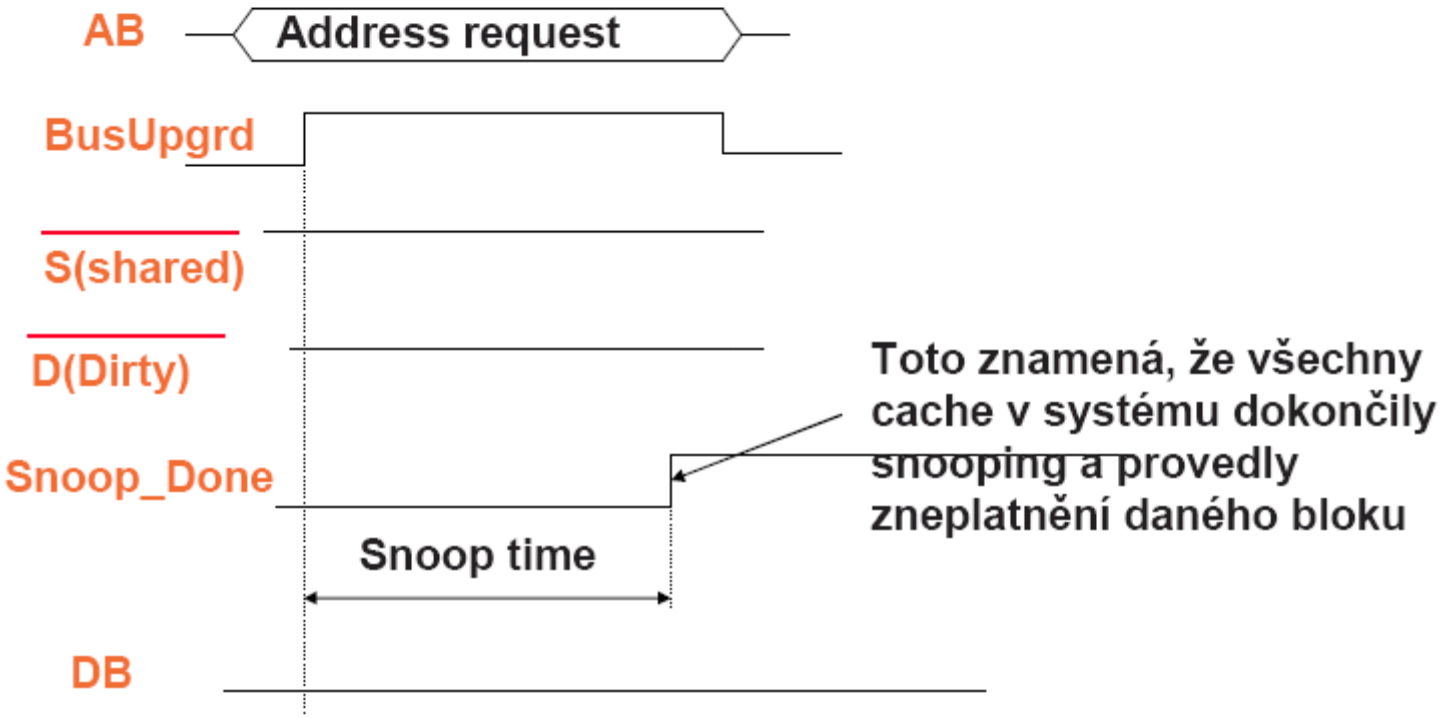
Příklad transakce – BusRd do *Shared*



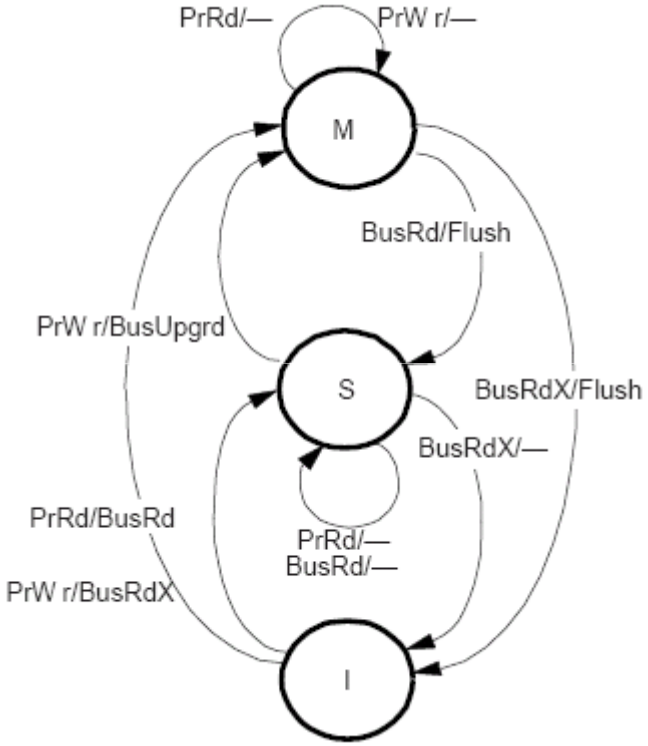
Příklad transakce – BusRd do *Shared z Modified*



Příklad transakce – BusUpgrd



MSI protokol

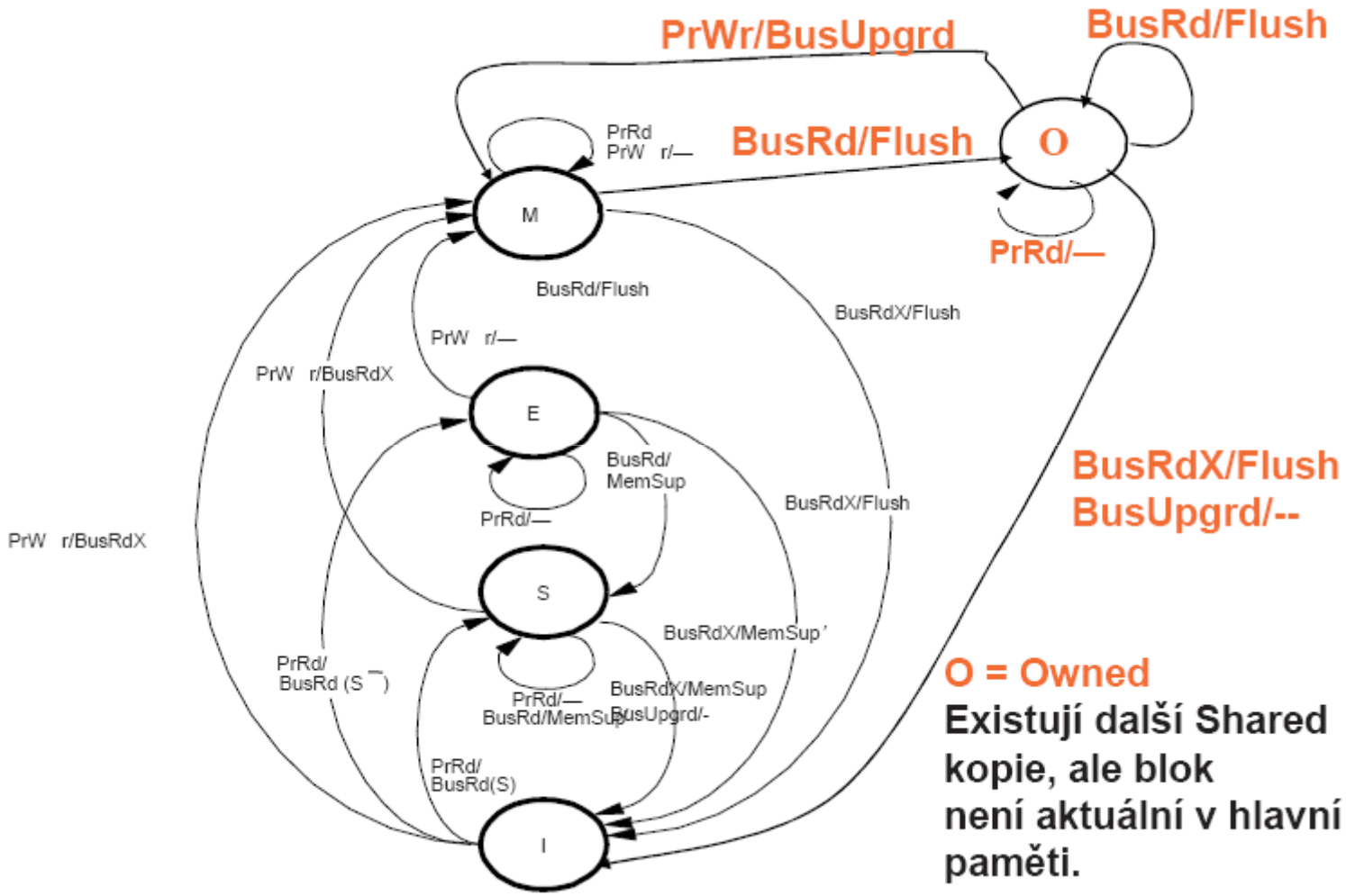


Chybí stav Exclusive,
není nutný signál S (shared)

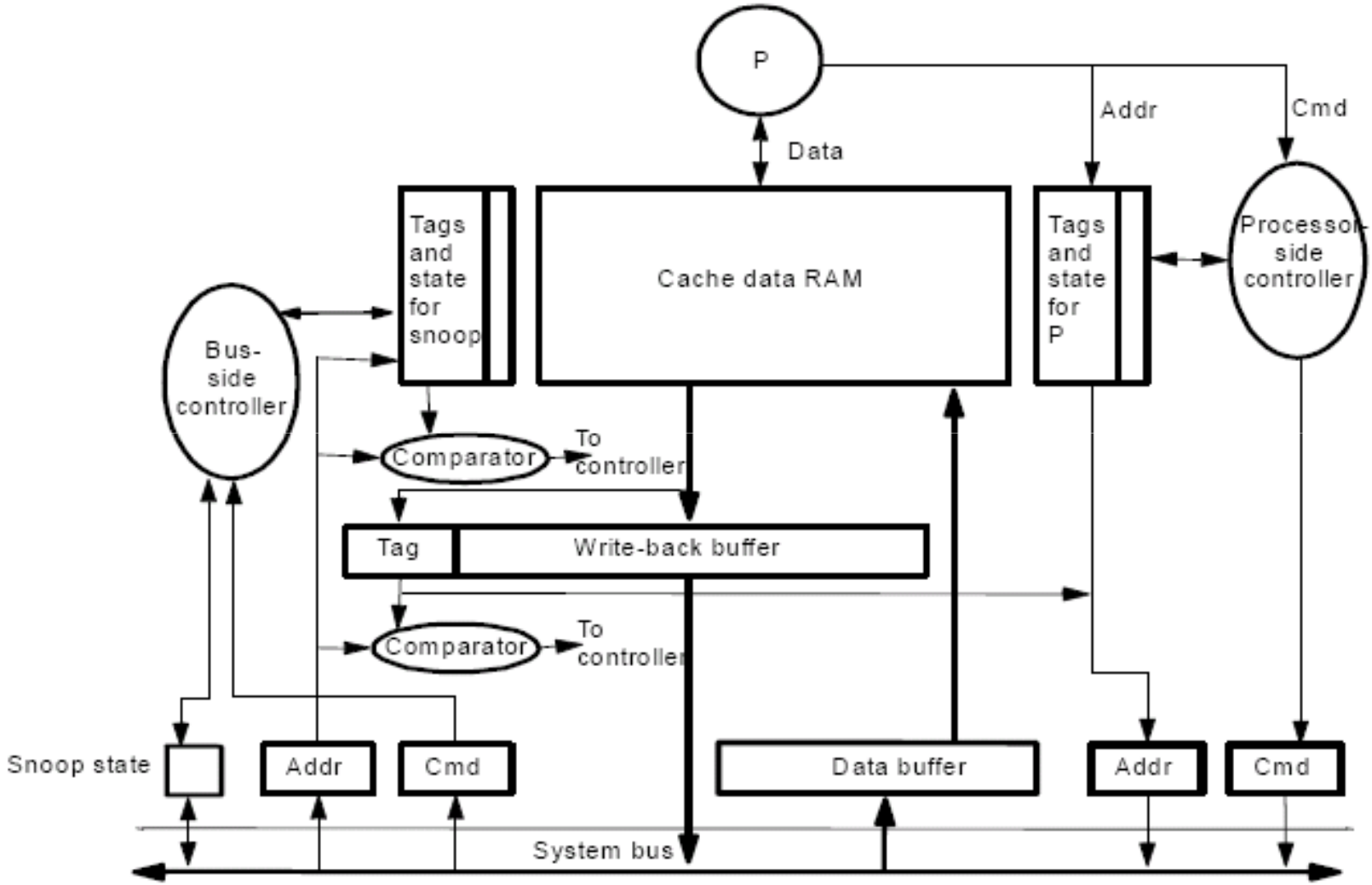
Před zápisem se vždy posílá
BusUpgrd nebo BusRdX

Výkonnost údajně jen o
10-20 % nižší než MESI

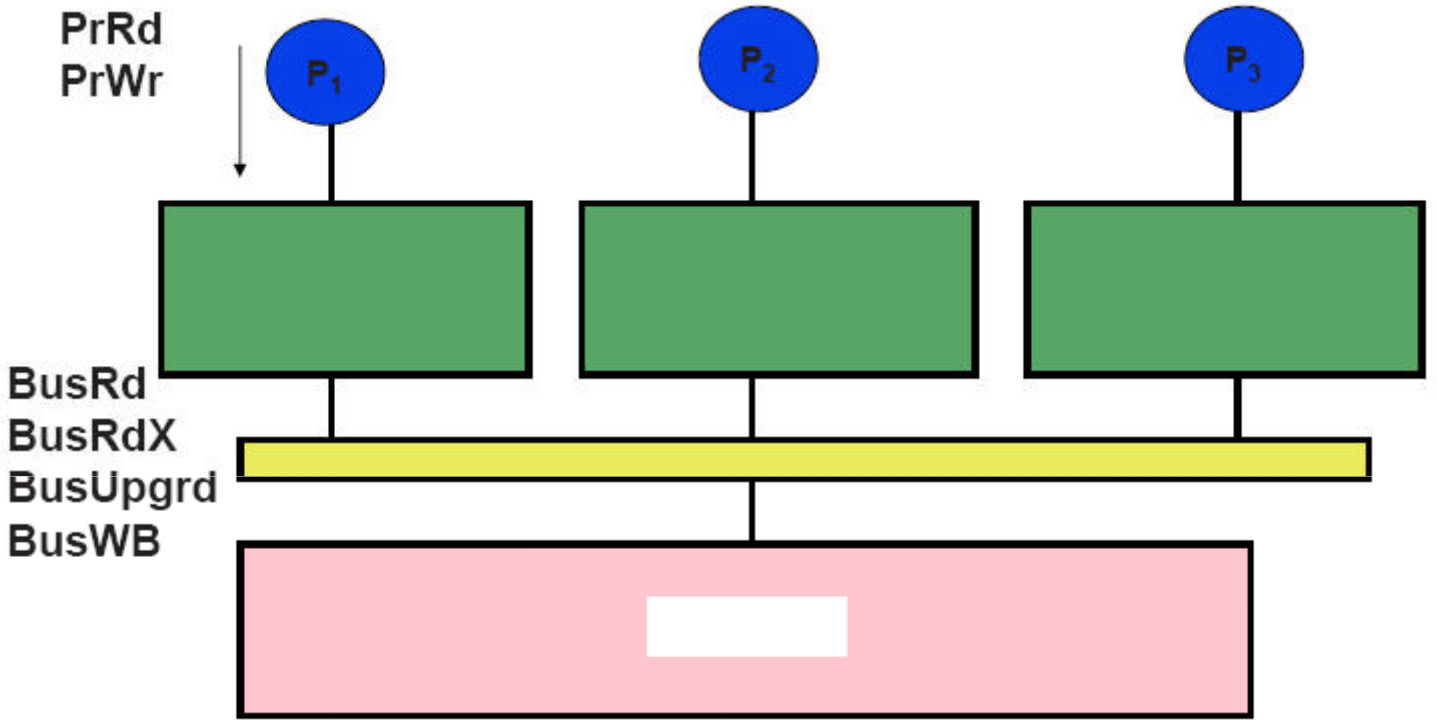
MOESI protokol



Řadič skryté paměti (zjednodušený)



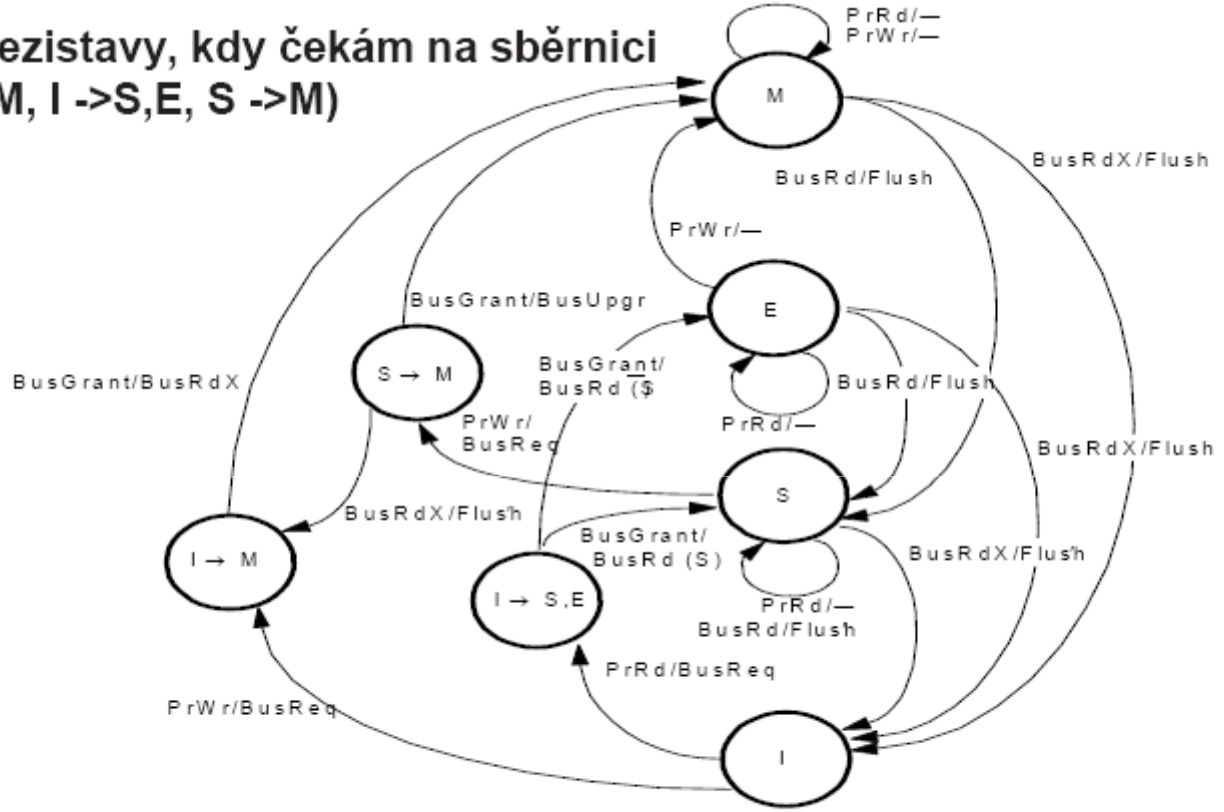
Implementace MESI na systému s atomickou sběrnicí



- Před spuštění sběrnicové transakce, je třeba o sběrnici žádat arbitr sběrnice (BusRq / BusGnt). Co když někdo mezitím provede transakci s naším blokem ?

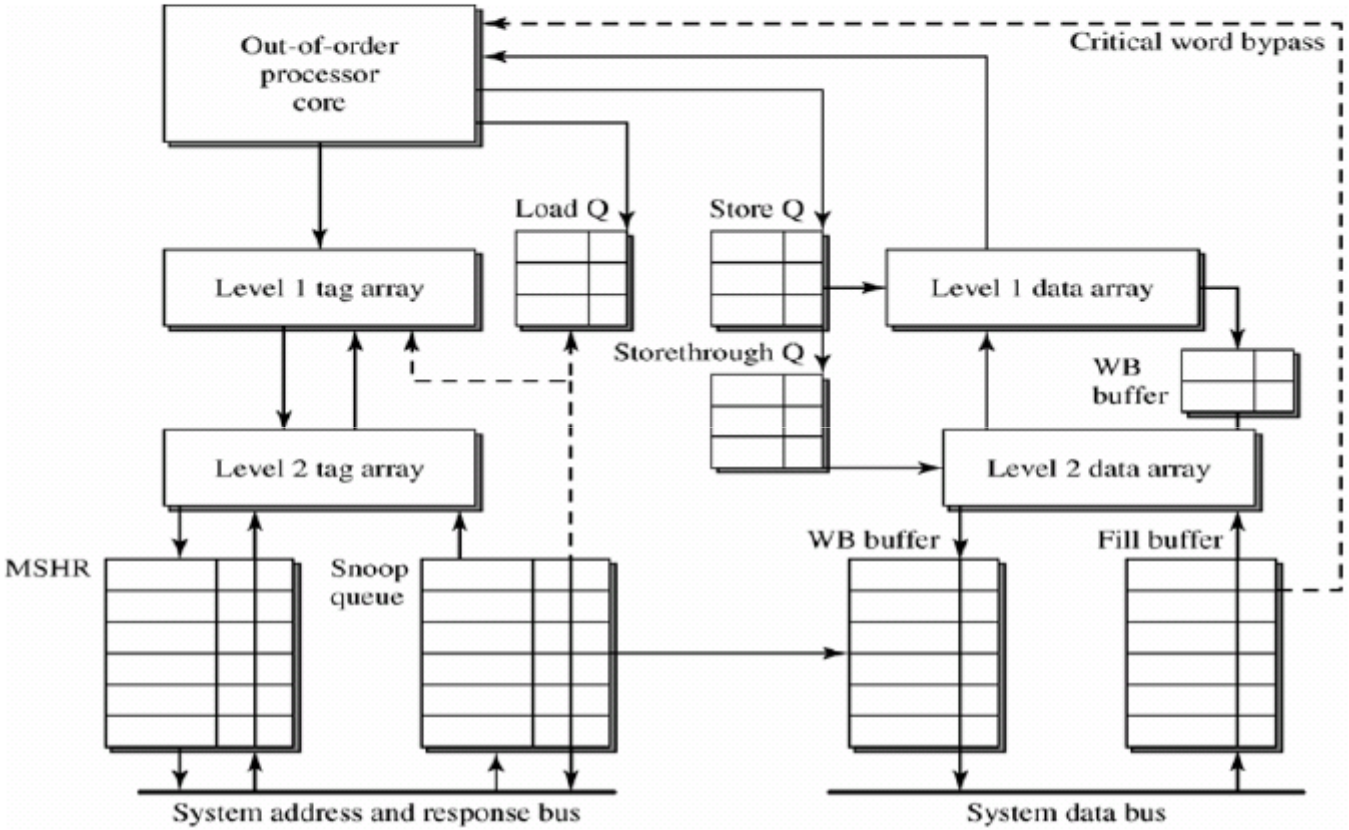
Řešení neatomického přechodu mezi stavy

3 mezistavy, kdy čekám na sběrnici (I->M, I ->S,E, S ->M)



Zajímavá situace: Chci z Shared do Modified, ve stavu S->M jiný procesor provedl BusUpgrd nebo BusRdx – musím *„zneplatnit“ blok* a přejít do stavu I->M a provést BusRdX místo BusUpgrd.

Reálný obrázek – víceúrovňový cache systém



A processor may communicate with memory through two levels of cache, a load queue, store queue, store-through queue (needed if L1 is write-through), MSHR (miss-status handling registers), snoop queue, fill buffers, and write-back buffers. Not shown is the complex control logic that coordinates all this activity.

Larger MPs

- Separate Memory per Processor – but sharing the same address space – Distributed Shared Memory (DSM)
- Provides shared memory paradigm with scalability
- Local or Remote access via memory management unit (TLB) – All TLBs map to the same address
- Access to remote memory through the network, called Interconnection Network (IN)
- Access to local memory takes less time compared to remote memory – Keep frequently used programs and data in local memory? Good memory allocation problem
- Access to different remote memories takes different times depending on where they are located – Non-Uniform Memory Access (NUMA) machines

Directories

- Pokud broadcast (multicast) nelze snadno realizovat (sběrnice)
- Základní myšlenka: Mít adresář (Directory), který indikuje pro každý řádek paměti:
  - zda je v cache (alespoň v jedné)
  - ve které cache/ích se nachází
  - zda je v cache dirty nebo clean
- **Full directory** obsahuje kompletní informace pro každý řádek paměti. Pro n procesorový systém, Boolovský vektor délky n+1. Pokud bit i (i=1,2,…n) je nastaven, i-ta cache má kopii řádky z paměti. Bit 0 pak indikuje zda je clean nebo dirty (v tom případě jenom jeden další bit může být nastaven = řádek je jenom v jedné cache)

Directories

- V NUMA systému, každý uzel má jenom část adresáře obsahující informace o řádcích uložených v jeho paměti = **home node**, ostatní: **remote node**
- Při cache miss, požadavek je poslán do domovského uzlu
- Full directory – nevýhodou je velikost adresáře. Například pro 8 procesorový systém kde velikost L2 cache řádků je 64 B (předpokládáme, že koherence je aplikována na úrovni L2) bude 2% (9/(64*8)=0.18), ale pro 64 procesorů 13%.
- Jiný extrém: stačí dva bity – je/není (Invalide) v cache, clean (Shared)/dirty(Modified) a Exclusivní – rozdíl je v tom, že stavy jsou kódovány v adresáři, ne v cache

IBM Power4 procesor – o něm někdy příště..