СИМЕТРИЧНИ И CC-NUMA МУЛТИПРОЦЕСОРИ

SYMMETRIC AND CC-NUMA MULTIPROCESSORS

Свойства на йерархията на паметта

- *Определят взаимоотношенията между съседните нива на паметта*
- *Определят функционирането на йерархията на паметта*

1. *INCLUSION PROPERTY (свойство на включването)*: **всички информационни обекти (думи, линии на кеша или страници) съхранени на ниво M_i представляват подмножество от обектите, съхранявани на ниво M_{i+1} (за всички нива)**

Най-външното ниво M_n съдържа цялата info, необходима за изпълнението на програмата. На по-ниските нива подмножествата на M_n се копират на ниво M_{n-1}, и т.н., докато необходимите инструкции & данни се прехвърлят към най-вътрешното ниво M_1,, готови за изпълнение от CPU

- Ако дадена info се намира на ниво $M_i \rightarrow$ нейни копия се съдържат на по-високите нива M_{i+1}, M_{i+2},..., M_n
- Обект в ниво M_{i+1} може да не се съдържа на ниво M_i т.е. на по-ниското ниво
- Липса на дума на дадено ниво задължително определя липсата на тази дума на всички по-ниски нива
- Най-високото ниво съдържа backup памет, която съдържа цялостната info

Data transfer units *– количеството info, което се обменя между съседни нива на паметта*: **думи** между регистрите на CPU & on-chip кеш и **линии между** вътрешния кеш & външния кеш M_1

2. СВОЙСТВОТО КОХЕРЕНТНОСТ

- ***ИЗИСКВА КОПИЯТА НА ЕДНА И СЪЩА INFO ДА БЪДАТ ИДЕНТИЧНИ НА РАЗЛИЧНИТЕ НИВА ОТ ЙЕРАРХИЯТА НА ПАМЕТТА : ако дума или линия на кеша е модифицирана на по-ниско ниво, копията трябва да бъдат непосредствено актуализирани на всички по-високи нива на паметта***
- ***Поддържането на кохерентност между последователните нива е нетривиална задача, често изискваща допълнителни шинни цикли или увеличена латентност на паметта***
- **Две стратегии за поддържане на кохерентност – *write-through & write-back***

3. ЛОКАЛНОСТ НА ДОСТЪПИТЕ

- ***Temporal locality*** (по време)– тенденцията да се осъществи достъп до последните данни в близък бъдещ момент
- ***Spatial locality*** (пространствена локалност)– тенденцията даден процес да осъществи достъп до info единици, чиито адреси са близки
- ***Sequential locality*** (на последователността)– отношението на изпълнението на последователност от инструкции и обработката на последователност от данни към това на на последователни такива е грубо 5 : 1 в обикновена програма

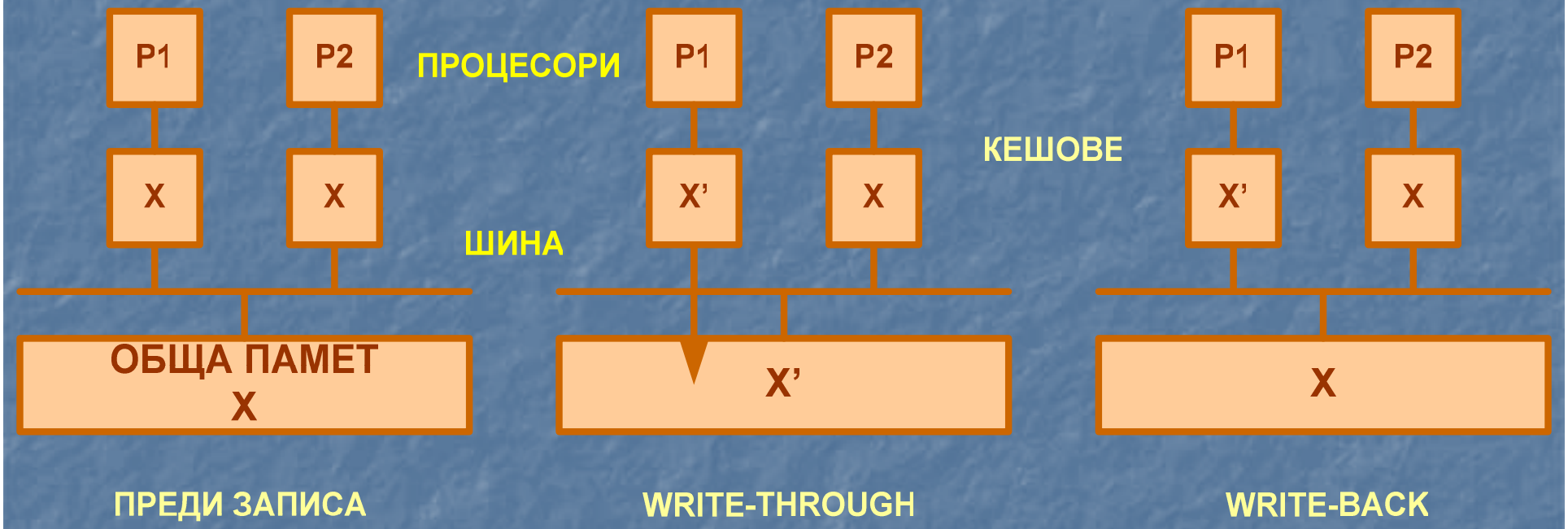
Проблемът за кохерентност на кешовете

- Отнася се до *различията в разпределените копия на една и съща линия , адресирани от общата памет*
- Четене, последвано от запис на X от процесор P, ще върне стойността, записана от P
- Четене от процесор P, последвано от запис на X от процесор Q, ще върне стойността, записана от процесор Q
- *Записите в една и съща линия на кеша X* от различни процесори *винаги са сериализирани* по такъв начин, че да определят един и същ ред на събитията в паметта, за всички процесори

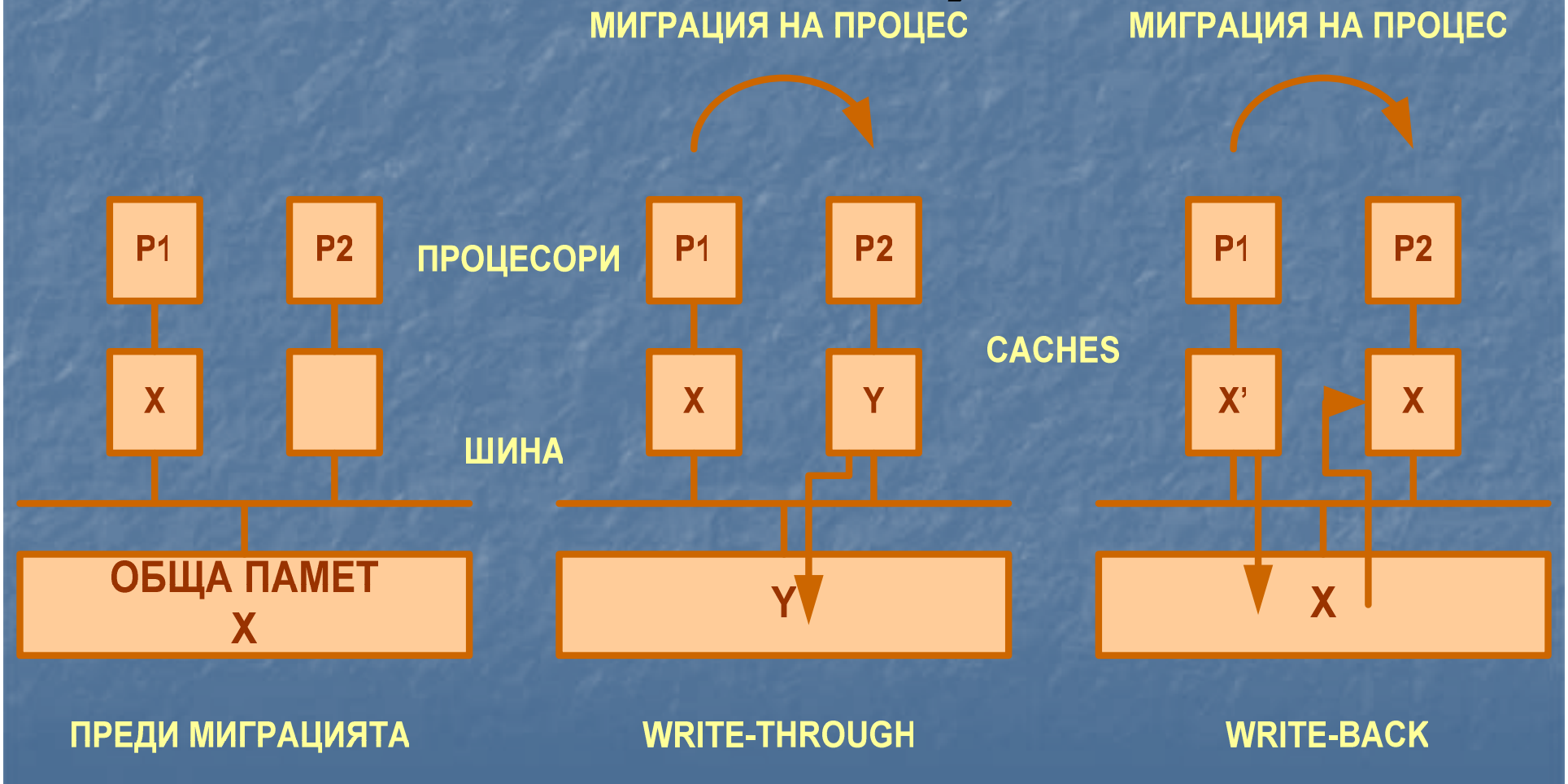
ИЗТОЧНИЦИ НА НЕКОХЕРЕНТНОСТ

1. ***Асинхронни записи*** на *различни процесори* в техните кеширани копия на една и съща линия на кеша в паметта
2. ***Миграцията на процеси*** между множество процесори без да се предупредят взаимно
3. ***В/И операции без участието*** на собствениците на кешираните копия

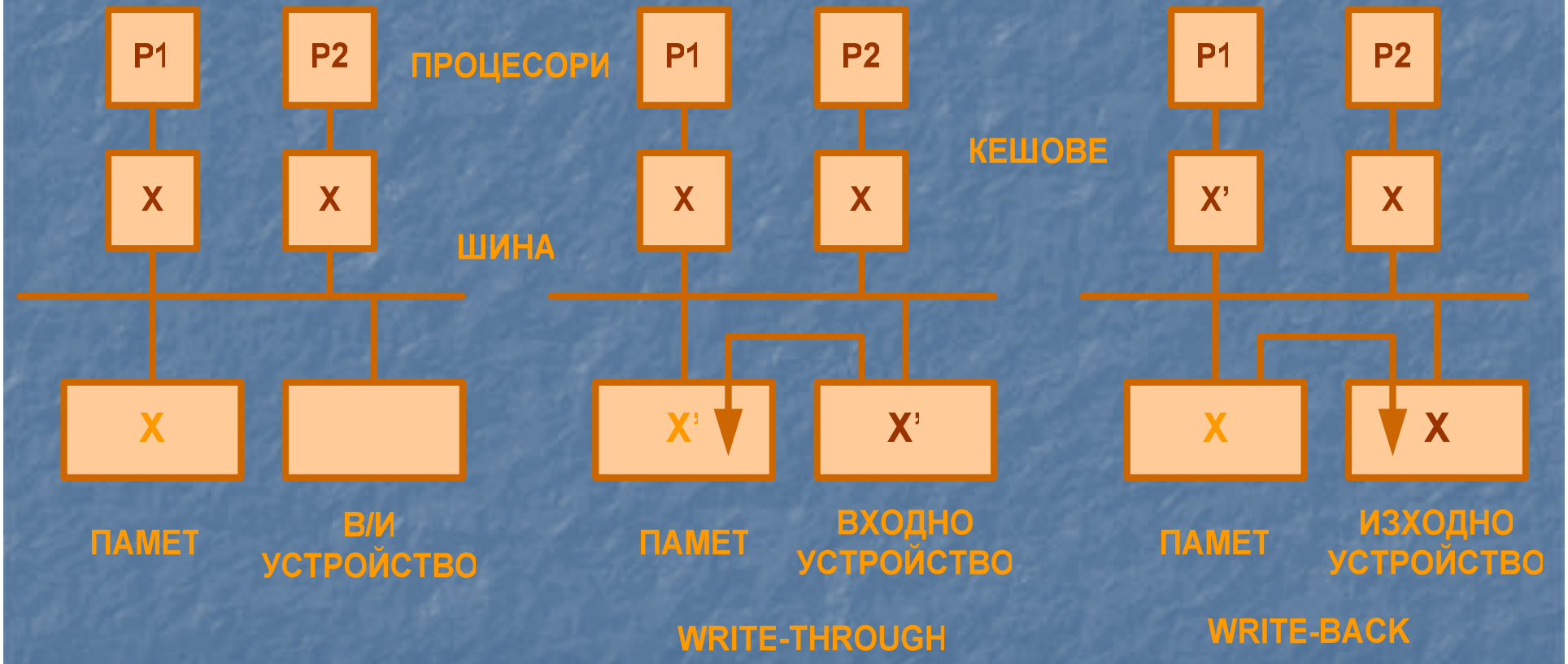
НЕКОХЕРЕНТНОСТ НА КЕША (ПОРАДИ ЗАПИС НА ОБЩА ПРОМЕНЛИВА)



НЕКОХЕРЕНТНОСТ НА КЕША (ПОРАДИ МИГРАЦИЯТА НА ПРОЦЕСИ)



НЕКОХЕРЕНТНОСТ, ПОРОДЕНА ОТ В/И ОПЕРАЦИИ БЕЗ УЧАСТИЕТО НА КЕША



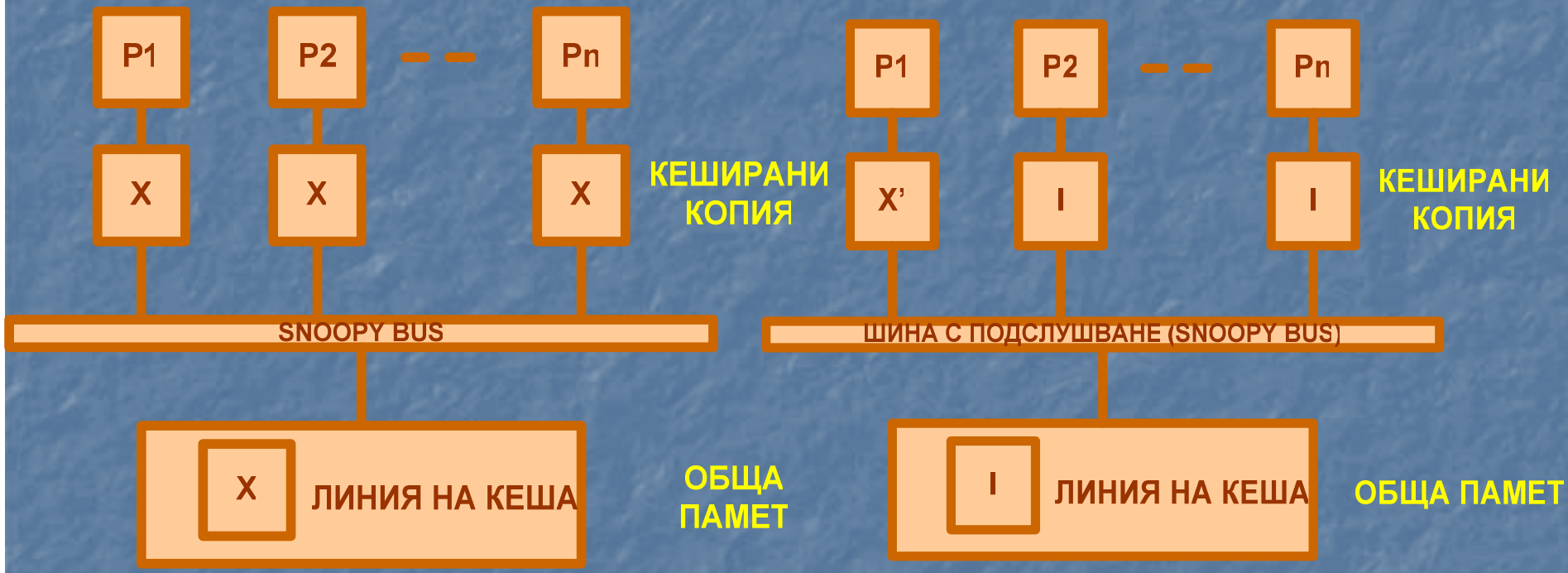
В/И ОПЕРАЦИИ ТРЯБВА ДА ИНФОРМИРАТ КОНТРОЛЕРА НА КЕША ЗА ПРОМЯНАТА!!!

SNOOPY COHERENCY PROTOCOLS
ПРОТОКОЛИ, ПОДДЪРЖАЩИ КОХЕРЕНТНОСТ С ПОДСЛУШВАНЕ

- Контрол на кохерентността се налага при запис на общи данни, миграцията на процеси, и В/И операции в мултипроцесорни платформи
- Два класа протоколи за кохерентност, имплементирани със *snoopy buses (шини с подслушване), следящи операциите, свързани с кеширането по шината)*
- ***Write – invalidate :*** прави невалидни всички други кеширани копия, когато се модифицира локално кеширано копие

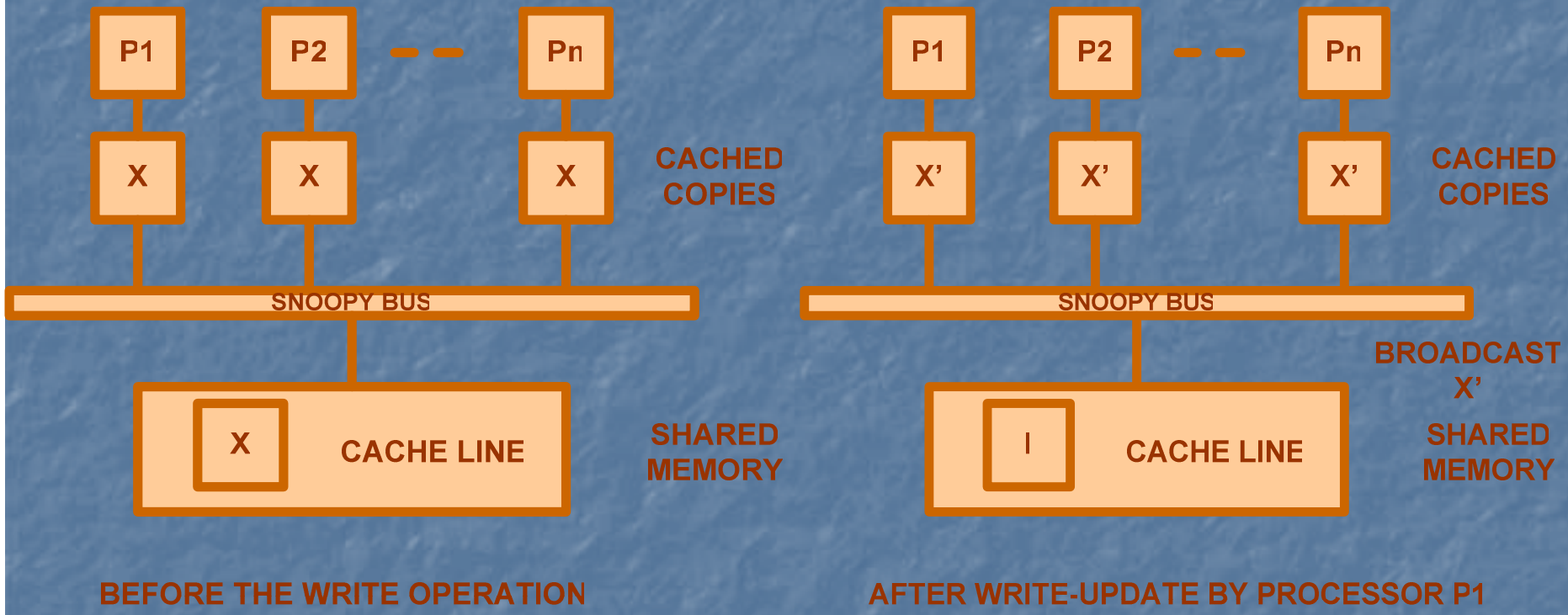
1. ***Write – update : изпраща (****broadcasts*) новото кеширано копие за актуализиране на всички други кеширани копия с един и същ адрес на линията

WRITE–INVALIDATE SNOOPY PROTOCOL



X' –modified cached line
I – cache line invalidated
WRITE-BACK CACHE

WRITE–UPDATE SNOOPY PROTOCOL



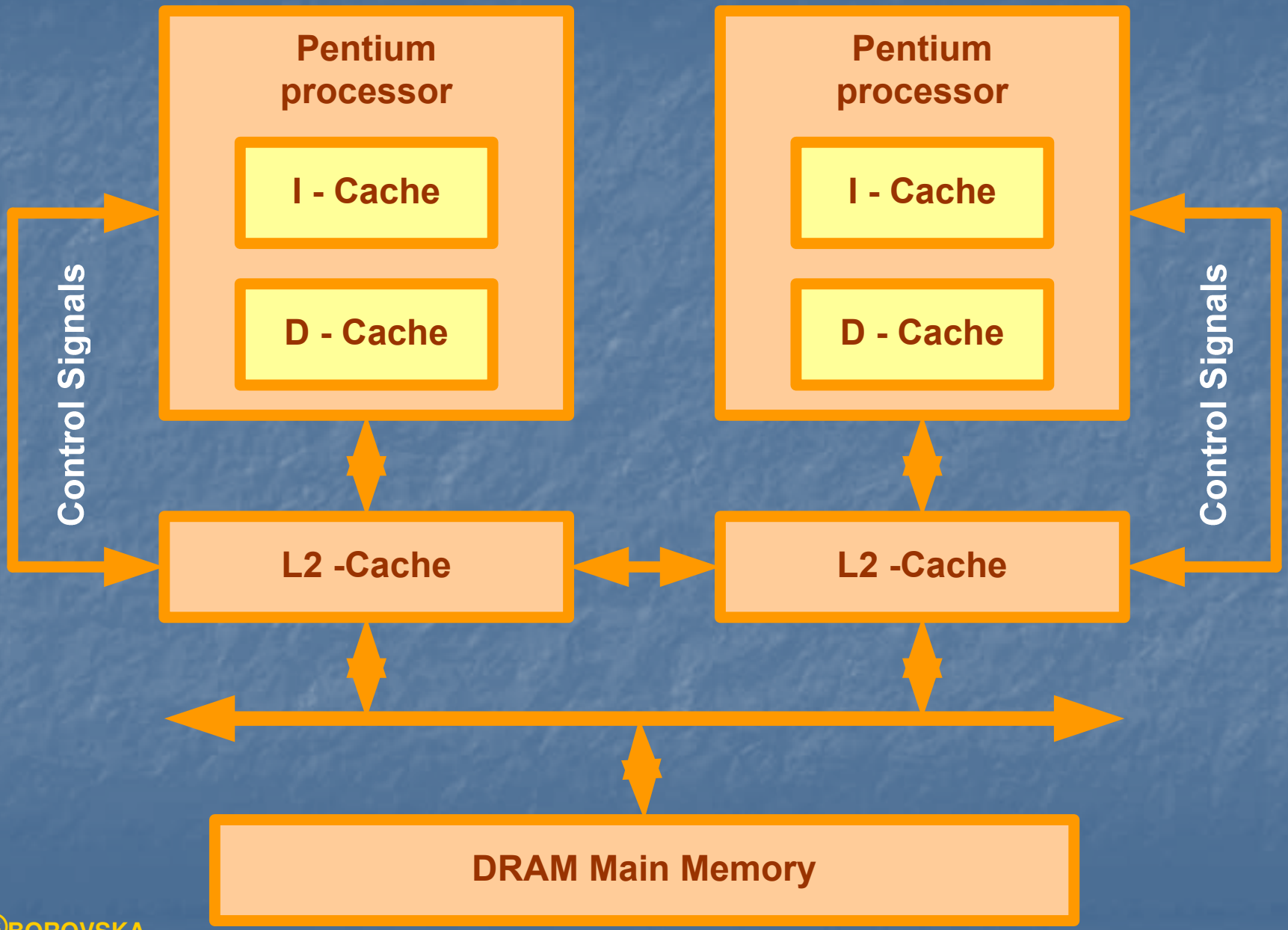
X' –modified cached line
I – cache line invalidated
WRITE-BACK CACHE

THE MESI SNOOPY PROTOCOL

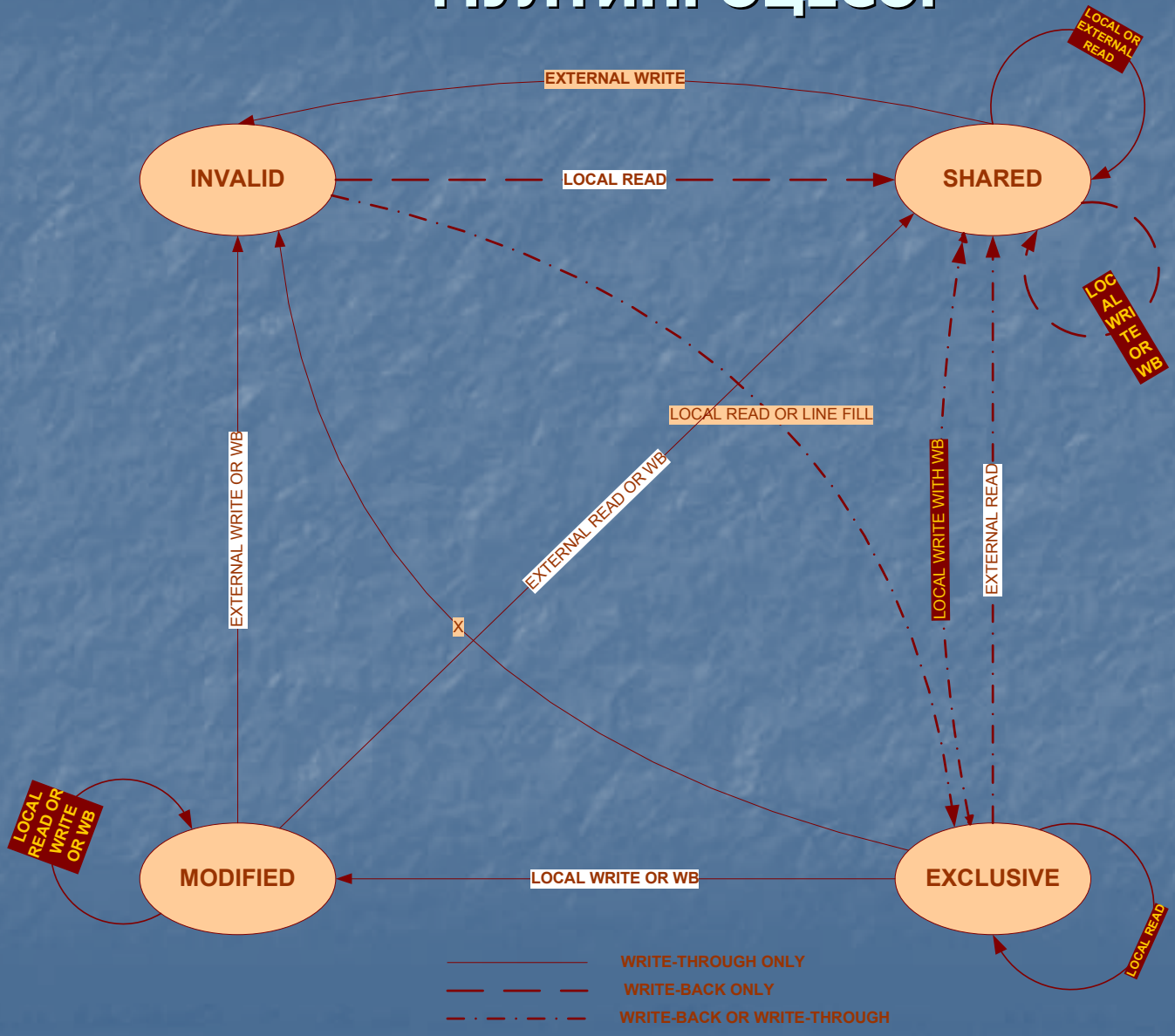
- MESI е *write-invalidate snoopy* protocol
- Следи състоянието *на всяка линия на кеша*, вкл. всички *read* или *write*, *cache hit* или *cache miss*, и *snoopy events* , детектирани по шината
- Pentium MESI protocol *поддържа както WB , така и WT кеширане,* управляван от външен сигнал
- *Всяка линия в кеша за данни* е винаги в едно от *4 възможни състояния*: Modified (M), Exclusive (E), Shared (S), Invalid (I)

- ***Modified (M state):*** линията на кеша е модифицирана с *write hit* от локалния процесор
- ***Exclusive (E state):*** съдържанието на линията в кеша е валидно, и невалидно във всички останали кешове. Съдържанието на паметта все още не е актуализирано.
- ***Shared (S state):*** линията е валидно, но също така тя може да бъда валидна в един или повече дистанционни кешове или в паметта
- ***Invalid (I state):*** Това е началното състояние след reset, или линията в кеша е станала невалидна от write hit в друг cache със същия адрес

A dual-Pentium multiprocessor with L2 caches



ГРАФ НА ПРЕХОДИТЕ ПРИ ПРОТОКОЛА MESI ЗА КЕША ЗА ДАННИ В PENTIUM-БАЗИРАН МУЛТИПРОЦЕСОР



ЛОГИЧЕСКА СТРУКТУРА НА СИСТЕМА С ОБЩА ПАМЕТ

МУЛТИПРОЦЕСОРИ

Системите с обща памет (SMP или CC-NUMA) имат следните ***предимства***:

- ***Симетрия*** : всеки процесор може да осъществи достъп до всяка клетка на паметта и до всяко В/И устройство
- ***Едно адресно пространство*** – има само едно копие на OS, базата данни, приложението, т.н., разположени в общата памет
- ***Кеширане*** – локалността на данните се поддържа от йерархията на кешовете
- ***Кохерентност*** – *поддържа се от hardware*
- ***Комуникация през паметта*** – *ниска латентност*, основана на прости load/store инструкции и hardware генерира info за кохерентността

СИМЕТРИЧНИ МУЛТИПРОЦЕСОРИ

- Повечето SMP's използват *шина*, като кохерентността на кешовете се осигурява от *MESI-like snoopy protocols*
  - Шинно-базираните SMP's са комерсиално успешни

<u>АСПЕКТИ НА SMP'S:</u>

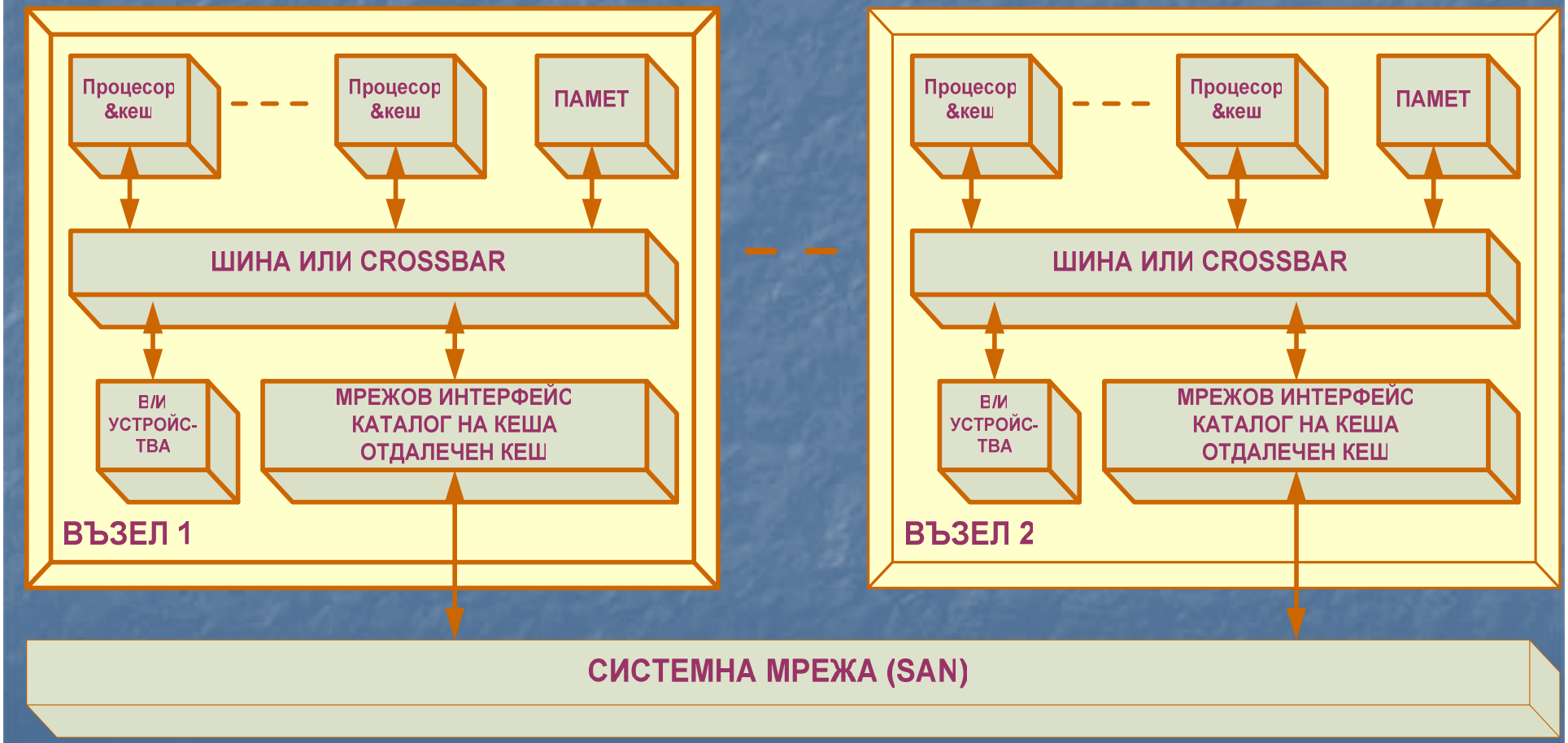
1. *Достъпност* – отказ на шината, паметта или OS, сриват цялата система
2. *Bottleneck* – шината на паметта & общата памет

3. *Латентност* – *ниска, сравнена с тази на мултикомпютрите*, стотици процесорни цикли
4. *Памет & В/И пропускателна способност* – за да се увеличат системната шина се замества с *crossbar switch network*, свързваща всички процесори с паметта & В/И устройства
5. *Мащабиране* – шината не подлежи на машабиране и ограничава броя на процесорите до десетки;

За мащабирането на по-големи системи се използват 3 подхода:

- crossbar
- CC-NUMA архитектура
- клъстериране

CC-NUMA АРХИТЕКТУРА
CC – Cache Coherent
NUMA – Non-Unified Memory Access



CC-NUMA системи

- *няколко SMP възли са свързани в по-голяма система*
- Използват протокол за поддържане на кохерентност с каталози
- Облекчава проблема за мащабирането на конвенционалните SMP's
- Архитектурата с разпределена обща памет *подобрява възможностите за мащабиране*; капацитета на паметта & В/И параметри могат да се увеличат чрез прибавянето на още възли
- Системният hardware & software автоматично разпределят данните между възлите
  - По време на изпълнение на програмата хардуерът, поддържащ кохерентността на кешовете автоматично премества данните към възлите, които ги ползват

Проблемът с отдалеченото кеширане
THE REMOTE CACHING PROBLEM

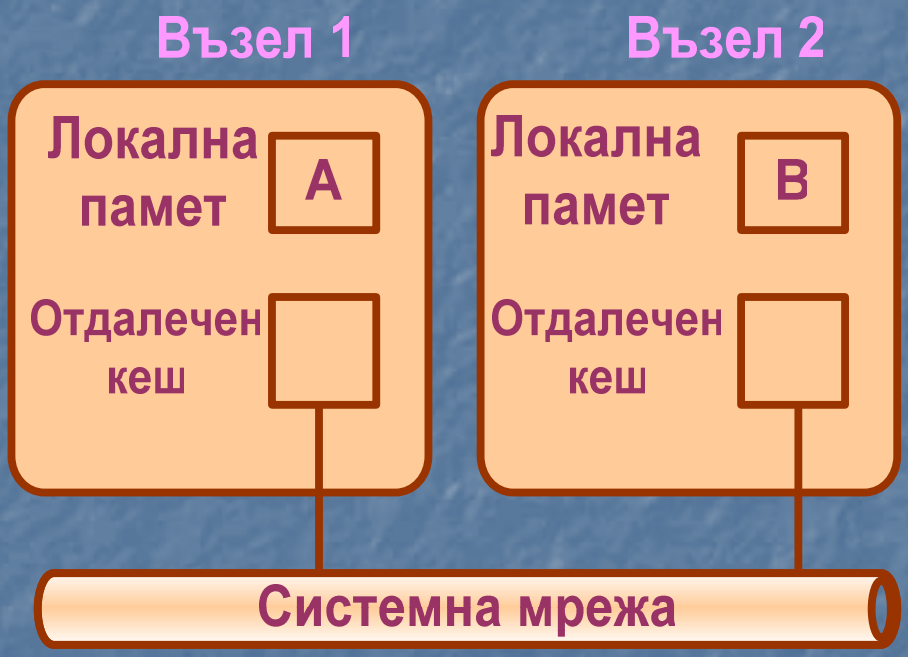
- *Отдалечените кешове* съдържат копия на данни, чиито оригинали са в отдалечена памет
- Нека да предположим, че дадена програма има два процеса P и Q, осъществяващи достъп до масивите от данни A и B:

Фаза 1: P използва (A) Q използва (B)

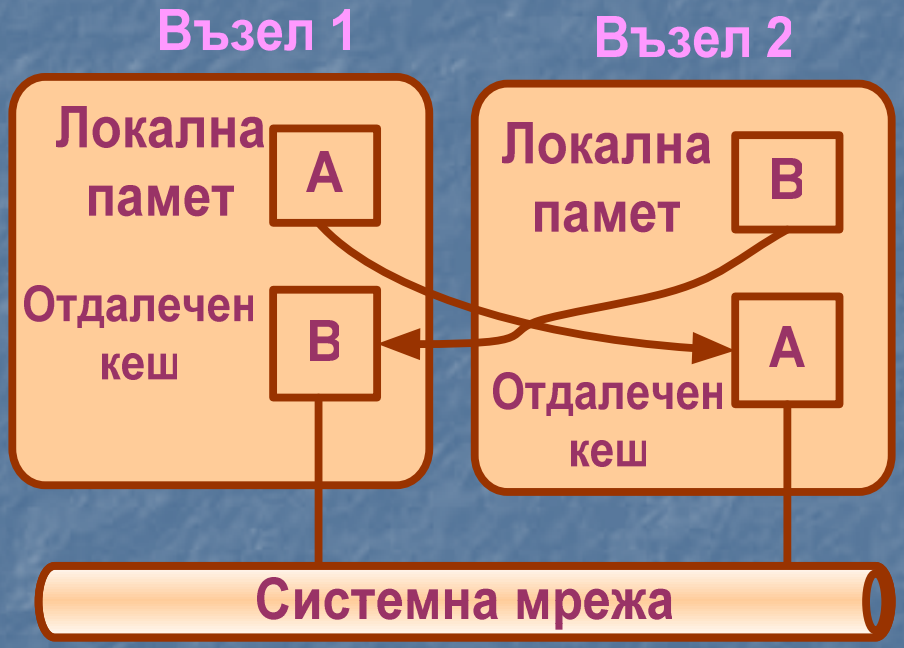
Фаза 2: P използва (B) Q използва (A)

Ако възникне недостиг на памет, съдържанието на отдалечените кешове трябва да бъде записано обратно в паметта на отдалечения възел.

ПРЕРАЗПРЕДЕЛЯНЕ НА ДАННИТЕ С ИЗПОЛЗВАНЕ НА ОТДАЛЕЧЕНО КЕШИРАНЕ (REMOTE CACHING) В CC-NUMA МУЛТИПРОЦЕСОР



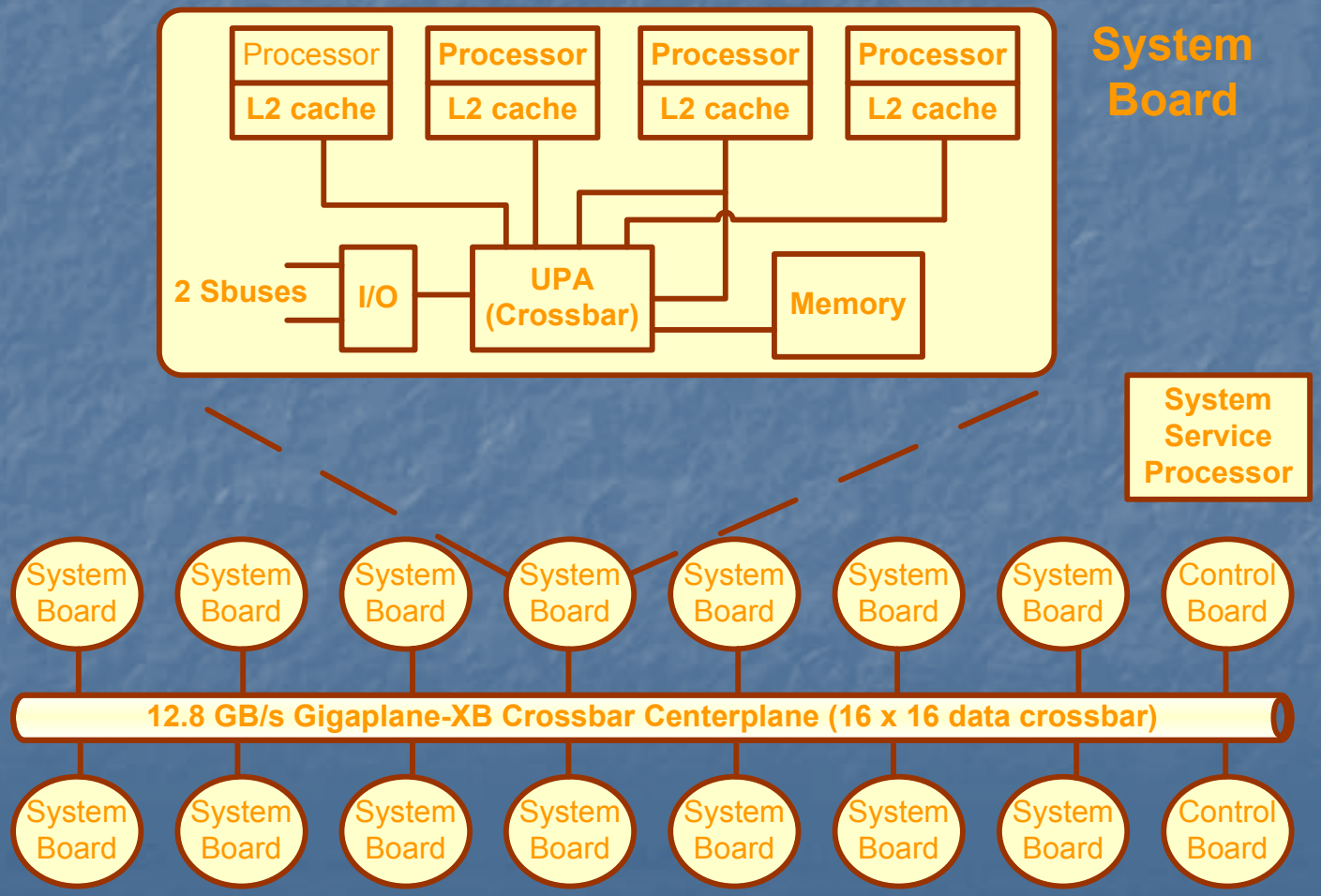
Разпределение на данните през фаза 1



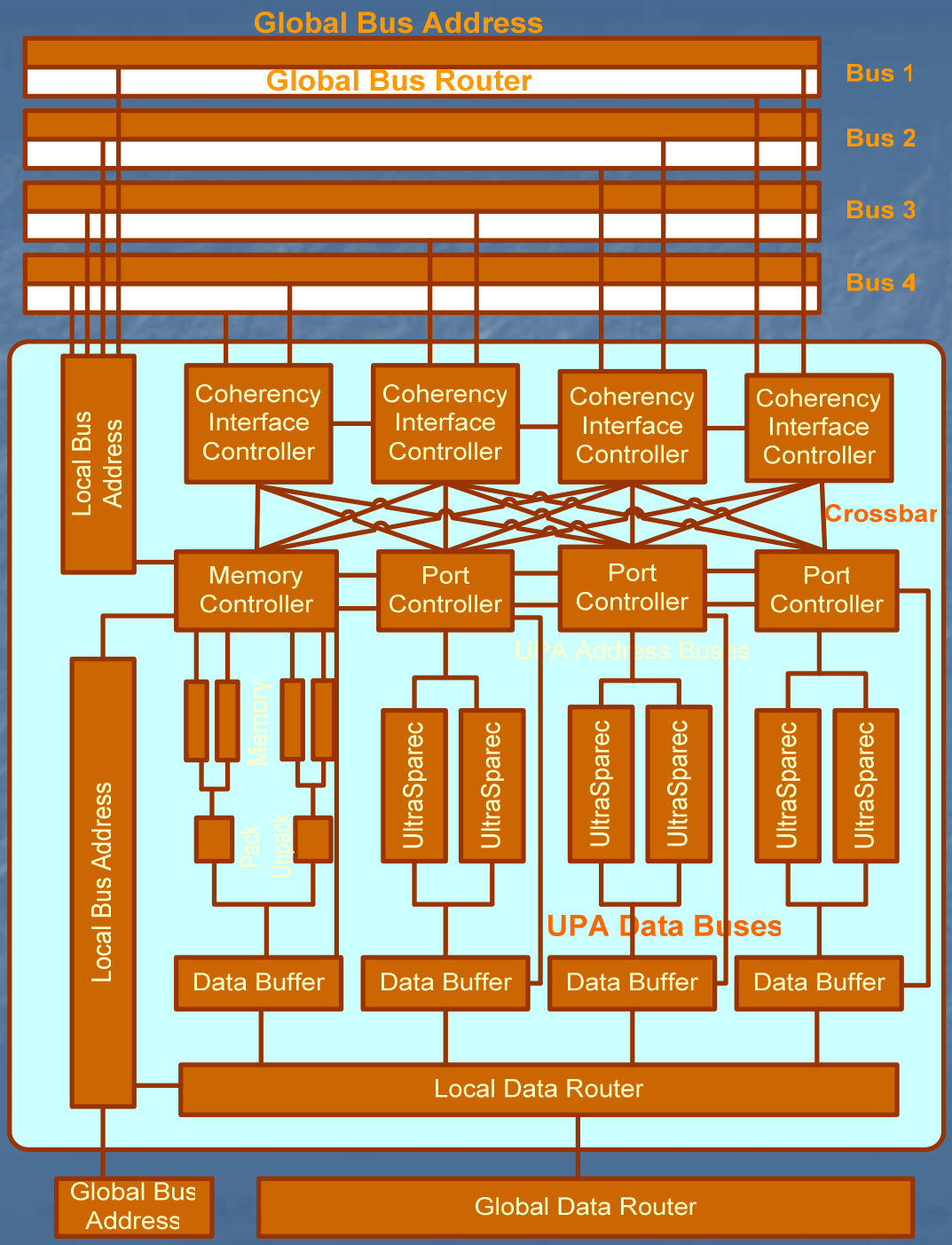
Разпределение на данните през фаза 2

КОМЕРСИАЛНИ SMP СЪРВЪРИ

- *High-end SMP server - **Sun's Ultra Enterprise 10 000***, наричан още ***StarFire***: 64 процесора, 64 GB памет, 20 TB on-line диск



Архитектура на системната платка на сървъра Sun StarFire



- ***Gigaplane-XB Interconnect*** – базиран на кросбар на две нива, 12.8 GB/s
- ***UPA bus*** - *Ultra Port Architecture (UPA) standard*, междинна шина, свързваща платките CPU/памет и В/И платки с Gigaplane bus, 128 b на порт, *snoop адресите се разпръскват към всички платки, пакетите с данни се изпращат point-to-point през crossbar switch*
- ***Global 16x16 data crossbar-*** маршрутизира пакетите с данни между 16 системни платки
- ***Маршрутизация на данните*** на две нива – global & local
- ***System service processor (SSP)*** – booting, shutdown, monitoring, system administration
- ***Processor module*** съдържа *UltraSPARC* microprocessor & 1 MB L2 cache

ПРОТОКОЛИ ЗА ПОДДЪРЖАНЕ НА КОХЕРЕНТНОСТ С КАТАЛОЗИ (DIRECTORY-BASED COHERENCE PROTOCOL)

- ***snoopy protocol*** се основава на възможността за *broadcast* на шината на паметта
- ***directory based-protocols*** не се основават на възможността за *broadcast* на шината на паметта

Cache directory

- ***Записва*** *местоположенията и състоянията* на всички кеширани линии на общите данни
- ***Указатели*** на всички отдалечени копия на всички кеширани линии
- ***A dirty bit*** – дали в даден кеш е променяно съдържанието
- ***Central directory*** – за управление на кохерентността на кешовете в малки SMP с централизирана обща памет

РАЗПРЕДЕЛЕНИ КАТАЛОЗИ

всеки модул памет поддържа отделен каталог

ТИПОВЕ КАТАЛОЖНИ ПРОТОКОЛИ

- ***ПЪЛЕН КАТАЛОГ НА КЕШОВЕТЕ***

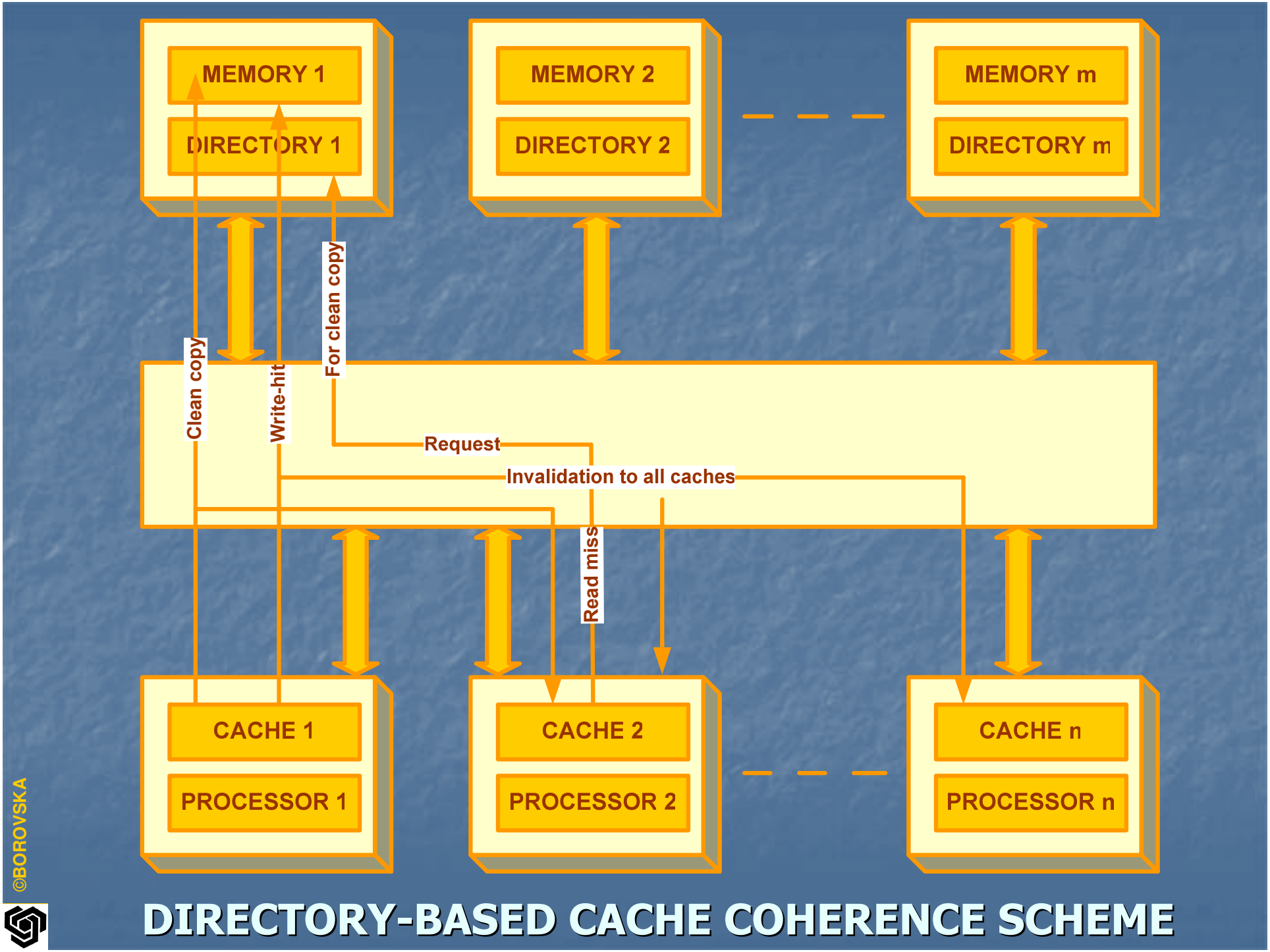
1. съдържа info за всички общи линии; всяка позиция на каталога съдържа N указателя, N- брой процесори
2. Всеки възел разполага с копие на пълния каталог, заема доста памет
3. Подходящ за малки мултипроцесори

- ***Ограничен каталог на кешовете***

1. Използва намален фиксиран # указатели за всяка позиция в каталога, независимо от размера на системата
2. Каталога заема по-малко памет
3. Може да забави актуализацията на кеш/памет

- ***Верижни каталози на кешовете***

1. Емулират схемата с пълен каталог чрез разпределяне на каталожната info в малки локални каталози
2. Необходимо е търсене през свързан лист от каталози (*IEEE SCI Standard* специфицира структурата на верижен протокол ***SCI Scalable cache coherence protocol)***

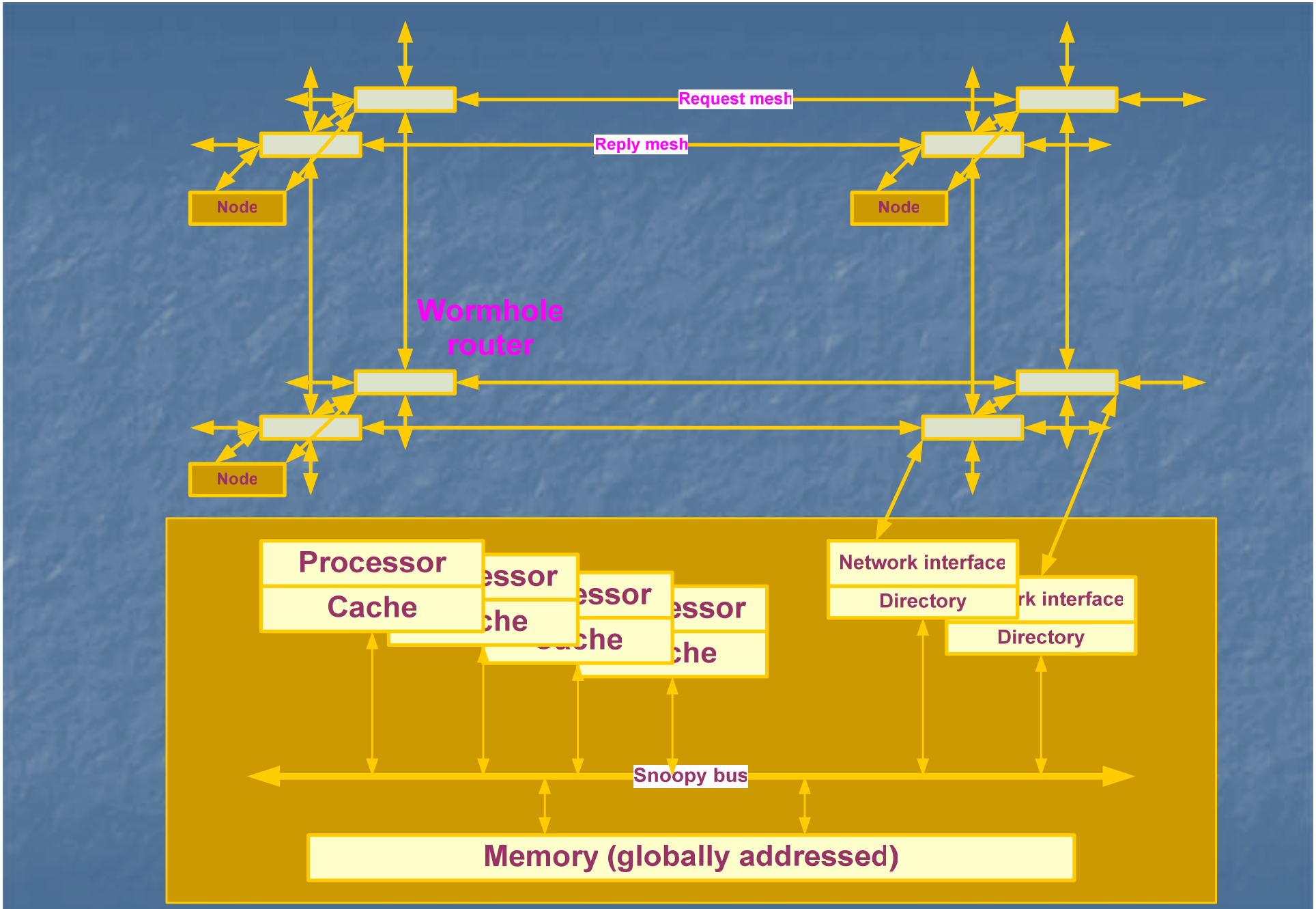

MEMORY 1
DIRECTORY 1
MEMORY 2
DIRECTORY 2
MEMORY m
DIRECTORY m
Clean copy
Write-hit
For clean copy
Request
Invalidation to all caches
Read miss
CACHE 1
PROCESSOR 1
CACHE 2
PROCESSOR 2
CACHE n
PROCESSOR n
DIRECTORY-BASED CACHE COHERENCE SCHEME

Stanford Dash:
a CC-NUMA multiprocessor project

- *Обща памет* – на ниво линия на кеш
- *Write-invalidation coherency protocol* с използване на разпределени каталози
- Кеш-линията в паметта или кешираното копие в локалните кешове може да бъде в 1 от 3 състояния:

1. *Uncached (некеширана)* – не е кеширана в никой възел на къстера
2. *Shared* – немодофицирани състояния в кешовете на 1 или повече възли на клъстера
3. *Dirty* – модифициран в един кеш на някой възел на клъстера

- *Directory* – съдържа обобщена информация за всяка линия (състояние и възлите на клъстера, които са я кеширали)



Архитектура на Dash

Йерархия на разпределената памет

КРАЙ