Лаб. Упражнение No: 9

Дисциплина: Компютърни Архитектури

Асистент: Иван Янчев

Тема: Кеш-кохерентни протоколи

========================================================================

**Теория:**

Кеш-кохерентни протоколи: Това са протоколи подсигуряващи еднаквост на данните в кешовете на различните процесори. Когато системата е изградена от повече от един процесора е нужно данните в кешовете да се синхронизират за коректно изпълнение на програмата. Всеки процесорен кеш може да притежава свое собствено копие от данните в паметта, като при тяхна промяна, той трябва да уведоми останали кешове и паметта. **Протокол MSI:**

Състояния: Modified, Shared, Invalid

**Invalid:** Данните в текущия кеш са стари или не са заредени в него.

**Shared:** Един или повече кешове имат валидни данни, съответстващи на тези в паметта.

**Modified:** Определеният кеш в това състояние е извършил писане (промяна на данните), което все още не е отразено в паметта и в другите кешове. В един момент от време е възможно само едно състояние M в само един от процесорните кешове.  
  
**Активна промяна на състоянията(процесорни):**  
Invalid -> Shared (при четене) // PrRd + BusRd

Invalid -> Modified (при писане) // PrWr + BusRdX

Shared -> Shared (при четене) // PrRd

Shared -> Modified (при писане) // PrWr + BusRdX

Modified -> Modified (при четене) // PrRd

Modified -> Modified (при писане) // PrWr

При активната промяна на състоянията кешовете променят състоянията на база на действие(четене или писане) извършено от самите тях. Освен това те са и тези които генерират сигнали по шината. Имената на сигналите са показани след // в таблицата.

**Сигнали:**

PrRd -> вътрешен за процесора сигнал read

PrWr -> вътрешен за процесора сигнал write

BusRd -> сигнал върху шината че е настъпил read (Ако има блок в състояние **M**odified, той трябва да запише данните си опратно в оперативната памет и да премине в състояние **S**hared)  
  
BusRdX -> сигнал върху шината че е настъпил write (когато се улови от друг процесор, той трябва да инвалидира(**I**) данните си за конкретният блок, ако притежава копие от него)

Flush -> Генерира се когато процесор записва блок от данни обратно в оперативната памет.

**Пасивна промяна на състоянията(шинни):**

Пасивната промяна на състоянията настъпва когато процесор прочете по шината сигнал подаден от друг процесор за блок който притежава. Тогава той трябва да обнови състоянието на своят блок.

Invalid + BusRd -> Invalid

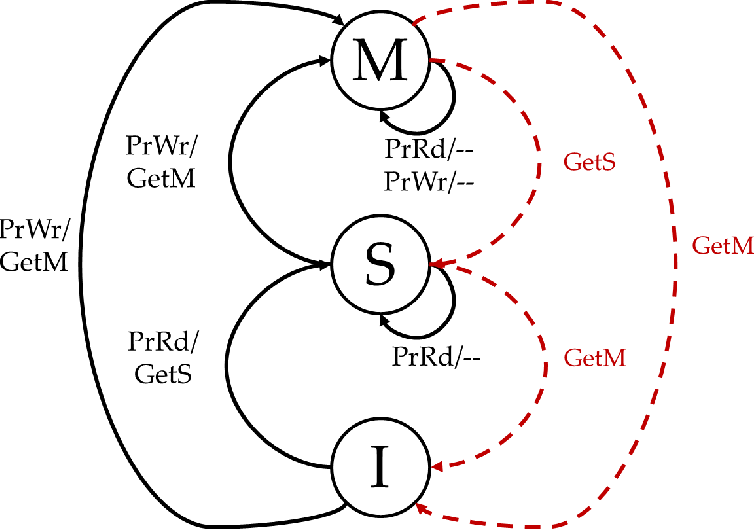
Invalid + BusRdX -> Invalid  
Shared + BusRd -> Shared

Shared + BusRdX -> Invalid

Modified + BusRd -> Shared + **Flush**

Modified + BusRdX -> Invalid + **Flush**

При преминаване от състояние **M**odified в което и да е друго състояние, данните от неговият кеш трябва да бъдат записани обратно в оперативната памет, от където други процесори да ги вземат. Генерира се сигнал **Flush** от пишещия процесор.



**Протокол MЕSI(Illinois):**

Състояния: Modified, Exclusive, Shared, Invalid

Разликата тук спрямо MSI е, че е добавено едно допълнително състояние **E**xclusive. Exclusive се проявява като Shared, но само между един кеш и оперативната памет. Ако повече от един кешове притежават валиден блок, то състоянието ще бъде Shared.

**Активна промяна на състоянията:**  
(not S) Invalid -> Exclusive (при четене) // PrRd + BusRd

(S) Invalid -> Shared (при четене) // PrRd + BusRd

Invalid -> Modified (при писане) // PrWr + BusRdX

Shared -> Shared (при четене) // PrRd

Shared -> Modified (при писане) // PrWr + BusRdX  
Exclusive -> Exclusive (при четене) // PrRd

Exclusive -> Modified (при писане) // PrWr

Modified -> Modified (при четене) // PrRd

Modified -> Modified (при писане) // PrWr

От тук се виждат ползите от допълнителното състояние. Ако Exclusive беше заменен със Shared, както е при MSI, то при всяко писане щеше да се генерира трафик по шината BusRdX, който е избегнат при Exclusive. Това може да не звучи като голямо подобрение, но тази операция се случва изключително често. Освен това в голяма част от случаите няма споделяне на един ѝ същи блок от данни между различните кешове, това зависи от програмата. В такъв случай генерирането на допълнителен трафик е напълно излишно.

**Сигнали:**

Сигналите са същите като при **MSI**.

**Пасивна промяна на състоянията:**

Invalid + BusRd -> Invalid

Invalid + BusRdX -> Invalid  
Shared + BusRd -> Shared + Flush\*

Shared + BusRdX -> Invalid + Flush\*

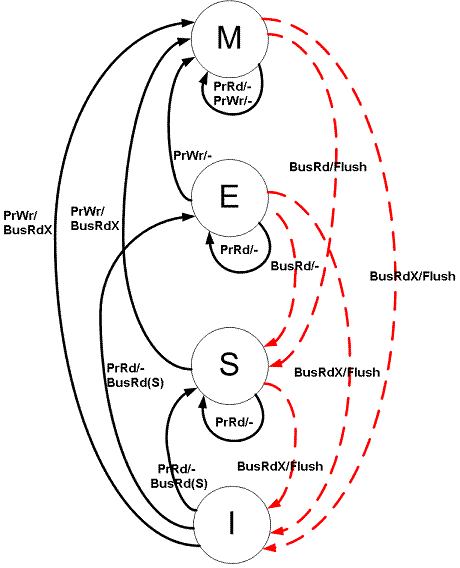
Exclusive + BusRd -> Shared

Exclusive + BusRdX -> Invalid

Modified + BusRd -> Shared + **Flush**

Modified + BusRdX -> Invalid + **Flush**

Понеже не е ефективно данните всеки път да се извличат от оперативната памет(защото е бавна), извличането им може да става и от чужд кеш. Това са червените Flush\* сигнали. На шината се подава цял кешов блок.



**Протокол Dragon:**

Предишните разглеждани протоколи бяха инвалидиращи (данните се инвалидират когато се обновят в друг кеш). Този протокол е обновяващ, което означава, че данните ще се обновяват, вместо да се инвалидират. Представя се добре когато писане в един кеш е последван от четене от други кешове на съшият блок.

Състояния: Exclusive, Shared clean, Shared modified, Modified

**Exclusive:** Само този кеш притежава данните за блока.

**Shared clean:** Блокът се намира в повече от един кешове и текущият блок не е последният, който е извършил запис.

**Shared modified:** Блокът се намира в повече от един кешове и текущият блок е последният, който е извършил запис. Този кеш също е отговорен за записване на блокът обратно в оперативната памет, когато бъде изхвърлен.

**Modified:** Само този кеш притежава данните за блока, но те са били модифицирани от запис.

**Активна промяна на състоянията:**  
 Exclusive -> Exclusive (при четене) // PrRd

Shared clean -> Shared clean (при четене) // PrRd

Modified clean -> Modified clean (при четене) // PrRd

Modified -> Modified (при четене) // PrRd

Exclusive -> Modified (при писане) // PrWr

(S) Shared clean -> Shared modified (при писане) // PrWr + BusUpd

(not S) Shared clean -> Shared modified (при писане) // PrWr + BusUpd

(S) Shared modified -> Shared modified (при писане) // PrWr + BusUpd

(not S) Shared modified -> modified (при писане) // PrWr + BusUpd

Modified -> Modified (при писане) // PrWr

Ако блок, който е бил споделен(S), престане да бъде, защото друг кеш е изхвърлил данните си. То състоянито на блока не се променя, но ще се промени по различенначин когато се извърши действие върху него, спрямо ако беше споделен. Затова Mc и Ms имат по два варианта.

**Сигнали:**

Нови сигнали: **BusUpd:** Обновява блоковете в чуждите кешове.  
 **PrRdMiss:** Когато кеш не притежава копие на блокът и има четене.

**PrWrMiss:** Когато кеш не притежава копие на блокът и има писане.

Когато **PrRdMiss** и **PrWrMiss** се наблюдават, блокът преминава в едно от следните състояния:

(not S) PrRdMiss -> Exclusive

(S) PrRdMiss -> Shared clean

(not S) PrWrMiss -> Modified

(S) PrWrMiss -> Modified clean

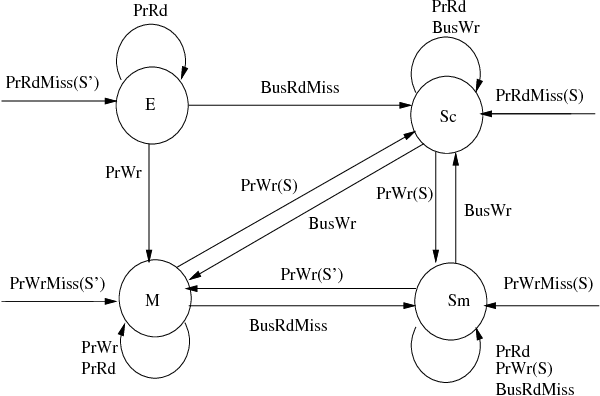
**Пасивна промяна на състоянията:**

Exclusive + BusRd -> Shared clean

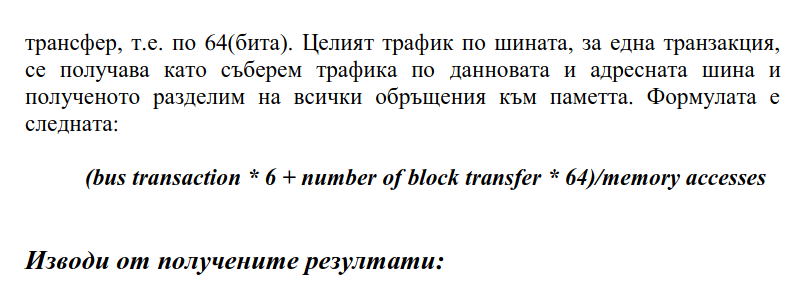
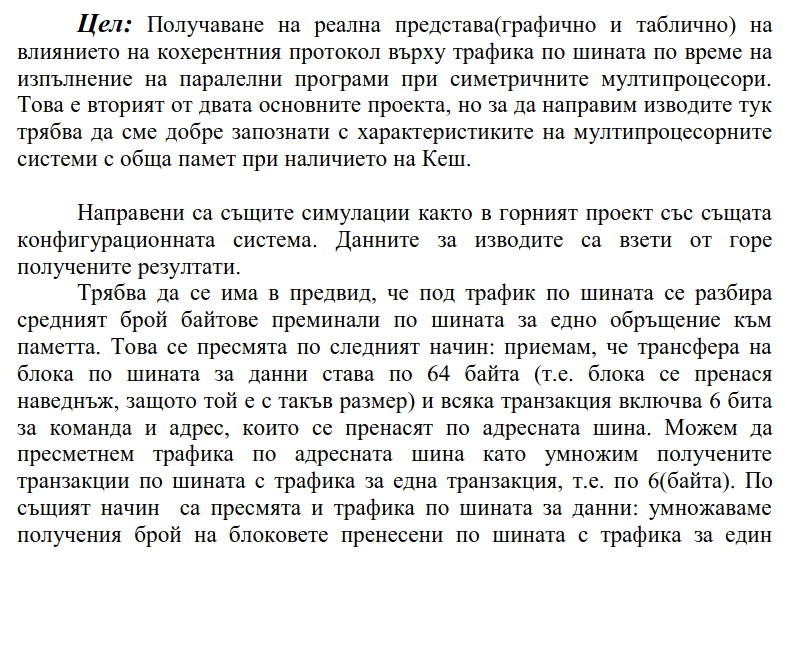
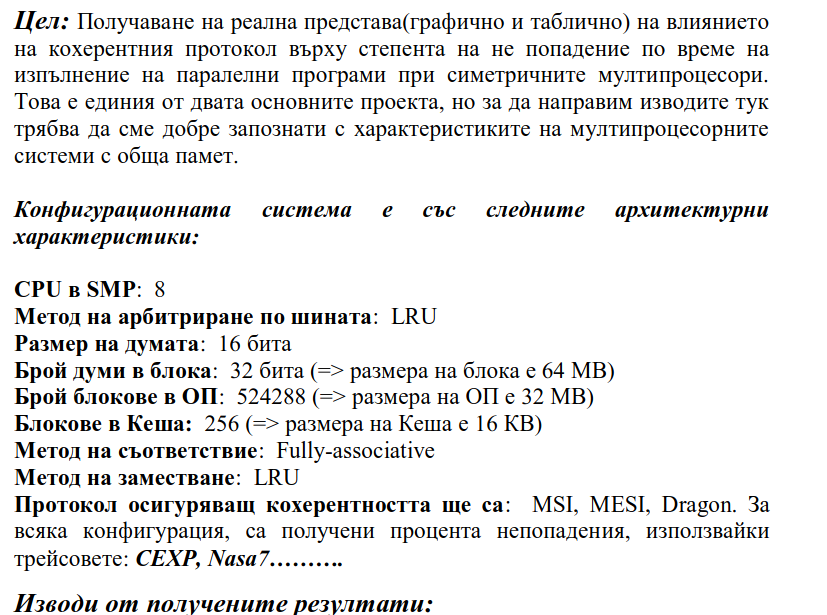
Modified + BusRd -> Shared modified

Shared modified + BusUpd -> Shared clean

Shared modified**/**Shared clean+ BusRd -> няма промяна.



1. **За протокола:**



Направете таблици с получените резултати. Напишете кратки изводи за влиянието на всеки от параметрите на кешовете общо за всички трейсове. Работата в екип е позволена за студенти от една ѝ съща група, ако си я разделят по трейсове, не по задачи. Нека всеки да си напише името до съответния трейс/група от трейсове, които е изпълнил.  
  
**Начин на работа:**  
1.Заредете трейсовете от **File > Open Memory Traces** > (Намерете папката на симулатора)**/Traces/PRG** > и кликнете на чекбоксовете отстрани (P1 – Pn).

2. Задайте архитектурата от трите менюта в **configure**.

3. Отидете на **Simulate** > **View cache evolution(text) > Execute.**

1. **Линкове**

**Упражнения**: [https://github.com/tu-iyan/Computer-Architecture-2024](./упражнения)

**Учебник:** [http://library.lol/main/373C67B0C5E22C9B92B1D8FACDC47E68](./учебник)

**Учебник 2:** <http://library.lol/main/4974CBF238F89810C9D70273B65BDADF>

**SMP\_Cache:** <https://dox.abv.bg/download?id=fd237b9f69#>

**Документация:** <http://arco.unex.es/smpcache/IICQTEI00.pdf>

**Оригинален линк:** <http://arco.unex.es/smpcache/>