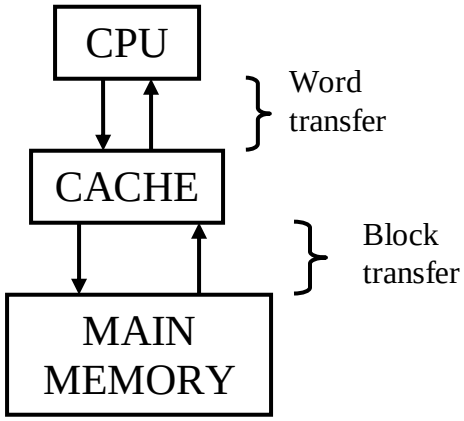
Упражнение 2

Изследване влиянието на размера на кеша и размера на блока върху степента на несъвпадение

I Теоретична част

Скоростта, с която може да се извърши достъп до информацията в оперативната памет е съществен фактор при проектиране на компютърните системи. Когато процесорът се обърне към оперативната памет за да извлече съответната инструкция трябва да чака, докато я получи, т. е. той губи време. Основен проблем при компютърните системи е разликата в скоростите на паметта и процесора. Един от начините, процесорът по-бързо да получава исканата информация от паметта, е използването на свръхбързодействащи памети, наречени кеш памети. Кеш паметта е изцяло хардуерно решение, т.е. прозрачно за програмата.



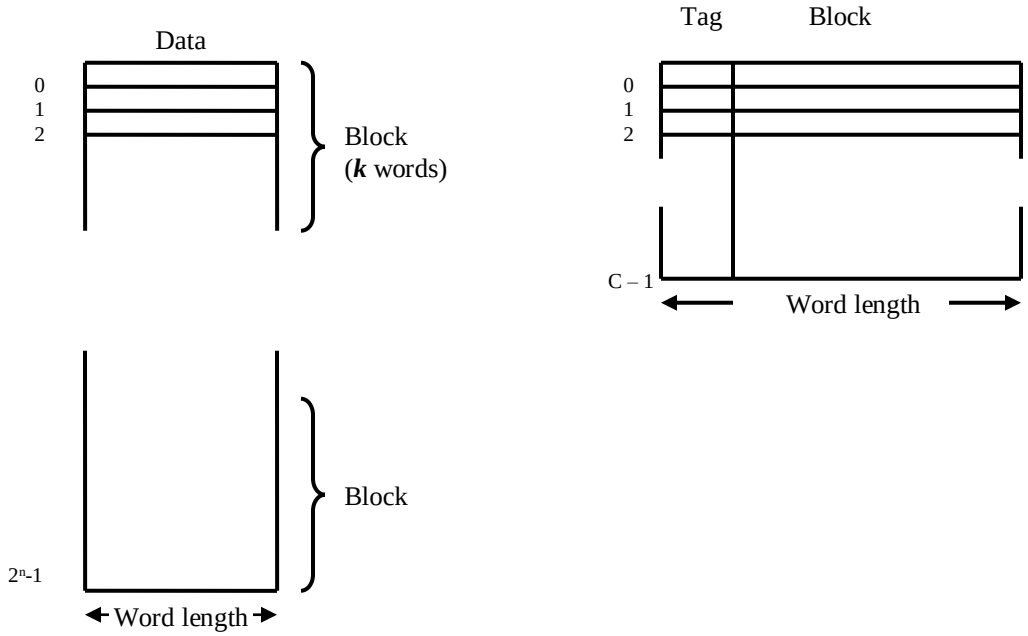
На фигурата са показани голяма по размер и бавна оперативна памет и по-малка по размер, но по-бърза кеш памет. Кешът се състои от високоскоростна памет с произволен достъп, която работи със скоростта, която се изисква от процесора.

В кеш се извършва трансфер на програмните инструкции и данни, които след това се предоставят на процесора. Обикновено оригиналните програми и данни се съхраняват в оперативната памет. Следователно информацията, която се намира в кеш е копие на част от информацията, която е разположена в оперативната памет. Адресът на даден блок от оперативната памет се съхранява по някакъв начин и в

кеш, тъй като процесорът ще се обърне към кеш със същия адрес от оперативната памет.

При операция четене, процесорът генерира адреса на думата от оперативната памет, която ще се прочете. Прави се проверка дали думата се намира в кеш. Ако да, тя се изтегля от кеш и се предоставя на процесора, ако не – се прави обръщение към оперативната памет, блокът съдържащ тази дума се зарежда в кеша и думата се предоставя на процесора. Необходимо е тази информация да се разположи в кеш, защото има вероятност след известно време тя да бъде необходима. Тази вероятност се основава на принципа на близостта във времето, който гласи, че ако сме се обърнали към някакъв адрес, вероятността да се обърнем към същия след известно време е много голяма**.**

Структурите на оперативната памет и на кеш паметта изглеждат по следния начин:



Оперативната памет се състои от определен брой блокове с фиксирана дължина, като всеки блок се състои от K на брой адресируеми думи, всяка от които има n - битов адрес. Така в оперативната памет има $M=2^n/K$ блока.

Кеша се състои от C на брой слота, всеки с по K думи, като във всеки слот има място само за един блок и от ляво на блока има етикет, който описва по някакъв начин принадлежността на този блок към оперативната памет. Броят на слотовете в кеша е по-малък от броя на блоковете в оперативната памет (C<M), тъй като само част от информацията от оперативната памет се намира в кеш.

Основание за успешното използване на кеша са някои характеристики на програмите.

Ако програмите се изпълняват последователно (в последователни адреси) и никога не се преизпълняваха едни и същи инструкции, щеше да е необходимо допълнително време, тъй като информацията трябва първо да се пренесе от оперативната памет в кеш и след това да се предостави на процесора. Тогава времето за достъп ще бъде:

Ts = Tc + Tm,

където Tc е времето за достъп до кеш паметта, а Tm е времето за достъп до оперативната памет.

Въпреки, че кода се изпълнява последователно, действително при изпълнение на всички програми многократно се извършват обръщения към едни и същи или близки адреси. Това се основава на *Принципа на близостта*, който е в сила както за обръщенията към данните, така и за обръщенията към инструкциите, въпреки че се среща по-често при инструкциите, отколкото при данните. Този принцип се проявява в два аспекта:

1/*Принцип на близостта във времето* – той гласи, че след като веднъж е извършено обръщение към някакъв адрес, вероятността да се извърши обръщение към същия в близкото бъдеще е много голяма;

2/ *Принцип на близостта в пространството* – след като е извършено обръщение към даден адрес, вероятността да се извърши обръщение към адреси близки на този е много голяма;

Въвежда се т. н. коефициент на попадение *h* и той се дефинира като вероятността дадена информация да се намира в кеш. Тази вероятност зависи от програмата и от размера и организацията на кеша. Обикновено попаденията трябва да са от 70 до 90%. Попадение имаме, когато търсената информация се намира в кеш. В противен случай се генерира непопадение и тогава е нужно да се извърши обръщение към оперативната памет.

Средното време за достъп се определя по следната формула:

Ts = Tc + (1 - h)*Tm, където:

Tc е времето за достъп до кеш. Тази част винаги участва във формулата, тъй като първоначално винаги се прави обръщение към кеш.

Tm е времето за достъп до оперативната памет;

Елементите на проектиране на кеша са размер на блока, размер на кеш паметта, методи за запис, методи на съответствие, методи на заместване, брой на кешовете. Всички те влияят върху производителността на компютърните системи.

Размер на кеша – Той трябва да бъде достатъчно малък, така че средната цена за един бит да е близка на тази при самостоятелна оперативната памет и достатъчно голям, така че средното време на достъп да е близко до това на система само с кеш. Има много други причини, заради които да се намали размера на кеша. По-големите кешове са по-бавни в сравнение с тези с по-малки размери. Друга причина, поради която е необходимо да се намали размера на кеша е това, че на чипа, на който е разположен процесора няма достатъчно място. Оптималния вариант за размер на кеш е между 1К и 512К думи. Но заради това, че производителността на кеша много зависи от натоварването на процесора workload, е невъзможно да се достигне до оптимален размер на кеша.

Размер на блока

Друг елемент свързан с проектирането на кеша е размера на блока или размера на реда. Когато се извърши трансфер на блок в кеш, заедно с желаната дума се пренасят и други съседни думи. С увеличаване размера на блока, коефициента на попадение първоначално нараства, което се основава на принципа на близостта в пространството. Освен това повече полезна информация се пренася в кеша. Коефициента на попадение започва да намалява обаче ако размера на блока продължи да се увеличава и вероятността да бъде необходима новопостъпилата информация е по-малка от вероятността за повторно използване на информацията, която трябва да се премахне. С увеличаване размера на блока броя на блоковете, които се намират в кеш намалява.

Връзката между размера на блока и коефициента на попадение е сложна и зависи от характеристиките на специфична програма и не е открита никаква оптимална стойност. Размер от 4 до 8 адресируеми единици (думи или байтове) е близък до оптимума.

Един от начините, по които може да се изчисли степента на несъвпадение е чрез използване на трейсове. Те представляват файлове, в които се съхраняват адресите от оперативната памет, към които ще се извършат обръщения. Тези трейсове се изпълняват на компютърна система, която може и да няма кеш, но задължително трябва да има процесор. Поддържа се запис на инструкциите и данните, към които се извършва обръщение. Симулират се специфични кеш организации като се използват обръщенията към данните и инструкциите, които са се извършили, за да се определи степента на несъвпадение. Обикновено са нужни стотици хиляди обръщения към адресите, за да се получат точни резултати.

Тъй като трейс програмите се изпълняват значително по-бавно отколкото самата тестваща програма, за провеждане на експериментите са нужни много часове.

1. Влияние на размера на кеша

Целта е да се покаже влиянието на размера на кеша върху степента на непопадение.

Задачи:

Конфигурирайте система със следните архитектурни характеристики:

- Брой процесори = 1;
- Протокол за кохерентност на кеша = MESI;
- Scheme for bus arbitration = Random;
- Дължина на думата (в битове) = 16;
- Брой думи в блок = 16 (размер на блока = 32 байта);
- Брой блокове в оперативната памет = 8192 (размер на оперативната памет = 256 KB);
- Метод на съответствие = Напълно асоциативно съответствие;
- Метод на заместване = LRU (метод на най-отдавна неизползвания);

Конфигурирайте блоковете в кеш използвайки следните конфигурации: 1 (размер на кеша = 0.03 KB), 2, 4, 8, 16, 32, 64, 128, 256 и 512 (размер на кеша = 16 KB). За всяка една конфигурация получете степента на несъвпадение използвайки трейс файловете (разширение ".prg"): *Hydro, Nasa7, Cexp, Mdljd, Ear, Comp,Wave, Swm и UComp.*

Въпроси:

Намалява ли или се увеличава степента на несъвпадение с увеличаване размера на кеша? Защо? Това намаление или увеличение случва ли се при изпълнение на всички трейсове или зависи от different locality grades? Какво се случва с capacity and conflict(collision) misses с увеличаване размера на кеша? Има ли conflict(collision) misses при този експеримент? Защо?

При този експеримент може да се наблюдава, че при големи размери на кеш паметта степента на несъвпадение се стабилизира. Защо? Също така можем да видим големи разлики в степента на несъвпадение при конкретно увеличение на размера на кеша. Какво показват тези големи разлики? Случват ли се at the same point при всички програми? Защо?

В заключение, увеличаването на размера на кеш паметта подобрява ли производителността на системата?

2. Влияние на размера на блока

Целта е да се изследва влиянието на размера на блока върху степента на несъвпадение.

Задачи:

Конфигурирайте система със следните архитектурни характеристики:

- Брой процесори = 1;
- Протокол за кохерентност на кеша = MESI;
- Scheme for bus arbitration = Random;
- Дължина на думата (в битове) = 16;
- Размер на оперативната памет = 256 KB (броя на блоковете в оперативната памет ще варива);
- Размер на кеш паметта = 4KB (броя на блоковете в кеш ще варира);
- Метод на съответствие = Напълно асоциативно съответствие;
- Метод на заместване = LRU (метод на най-отдавна неизползвания);

Конфигурирайте думите в кеш използвайки следните конфигурации: 4 (размер на блока = 8 байта), 8, 16, 32, 64, 128, 256, 512, 1024 (размер на блока = 2048 байта). За всяка една от тези конфигурации получете степента на несъвпадение използвайки трейс файловете: *Hydro, Nasa7, Cexp, Mdljd, Ear, Comp,Wave, Swm и UComp.*

Въпроси:

Степента на несъвпадение увеличава ли се или намалява с увеличаване размера на блока? Защо? Това увеличаване или намаляване случва ли се при всички програми или това зависи от different locality grades? Какво се случва с compulsory misses с увеличаване размера на блока? What is the pollution point? Проявява ли се при този експеримент?

В заключение, увеличаването на размера на блока подобрява ли производителността на системата?

3. Влияние на размера на блока при различни размери на кеш паметта

Целта е да се покаже влиянието на размера на блока върху степента на несъвпадение, но този път при различни размери на кеша.

Конфигурирайте система със следните архитектурни характеристики:

- Брой процесори = 1;
- Протокол за кохерентност на кеша = MESI;
- Scheme for bus arbitration = Random;
- Дължина на думата = 32;
- Размер на оперативната памет = 1024 KB (броя на блоковете в оперативната памет ще варират);
- Брой блокове в кеш = 128 (размер на кеш паметта = 4 KB);
- Метод на съответствие = Напълно асоциативно съответствие;
- Метод на заместване = LRU (метод на най-отдавна неизползвания);

Конфигурирайте думите в блок използвайки следните конфигурации: 8 (размер на блока = 32 байта), 16, 32, 64, 128, 256, 512 и 1024 (размер на блока = 4096 байта). За всяка една от тези конфигурации конфигурирайте броя на блоковете в кеш паметта за да получите следните размери за кеш: 4KB, 8 KB, 16 KB и 32 KB. За всяка конфигурация получете степента на несъвпадение използвайки трейса *Ear*.

Въпроси:

Първо ще ви зададем същите въпроси както при предишния проект: Степента на несъвпадение увеличава ли се или намалява с увеличаване размера на блока? Защо? Какво се случва с compulsory misses с увеличаване размера на блока? What is the pollution point? Проявява ли се при този експеримент?

Увеличава ли се или намалява влиянието на pollution point с увеличаване размера на кеша? Защо?

Упражнение 3

Изследване влиянието на методите на заместване и методите на съответствие върху степента на несъвпадение

I Теоретична част

Има различни начини, по които кеша може да бъде организиран, за да се съхранят данните. Във всички случаи процесорът се обръща към кеш с адреса, който идва от оперативната памет. Така, всяка кеш организация трябва да използва този адрес за да открие търсената информация от процесора, ако тя се намира там, или да уведоми процесора, в случай на непопадение. Проблемът за map на информацията, която се съдържа в оперативната памет в кеш трябва да се реализира хардуерно, за да се постигне подобрение при system operation. Възможни са различни стратегии.

Методи на съответствие

Става въпрос за съответствие на адрес от оперативната памет и мястото, където трябва да бъде той в кеш.

Има три метода на съответствие: метод на директно съответствие, метод на асоциативно съответствие и групово-асоциативно съответствие.

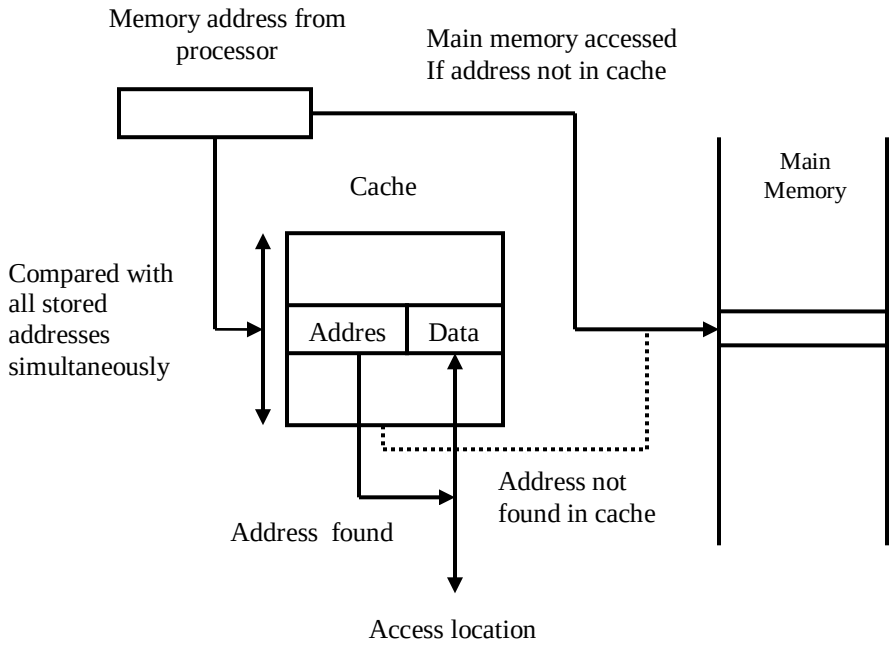
2.*Асоциативно съответствие*

Тук се преодоляват недостатъците на директното съответствие, тъй като даден ред от оперативната памет може да се разположи в който и да е ред от кеша. При този метод адреса от процесора не се разделя на части, а се използва целия адрес. Информацията в кеша се състои от две части: адрес и данни (фиг. 7). Този кеш се изпълнява като асоциативна памет, която съдържа както адреса на данните от оперативната памет, така и самите данни за всеки ред от кеш. Асоциативна памет е такава, при която достъп до информацията се осъществява не по адрес, а по съдържание. Проверява се дали има клетка в кеш, в лявата част на която да има адрес като този, който се получава от процесора. Тогава асоциативната памет проверява всички клетки в един такт, т. е. това е бърза памет. Адреса от процесора се сравнява с всички записани адреси едновременно. Ако има съвпадение данните се насочват към процесора, ако няма-отиваме в оперативната памет, информацията се чете от там и се предоставя на процесора. Тази информация трябва да се разположи в кеш, с цел по-нататък да се използва отново.

Възможно е информацията да се разположи в кеш и да се предостави на процесора едновременно. При запис информацията в

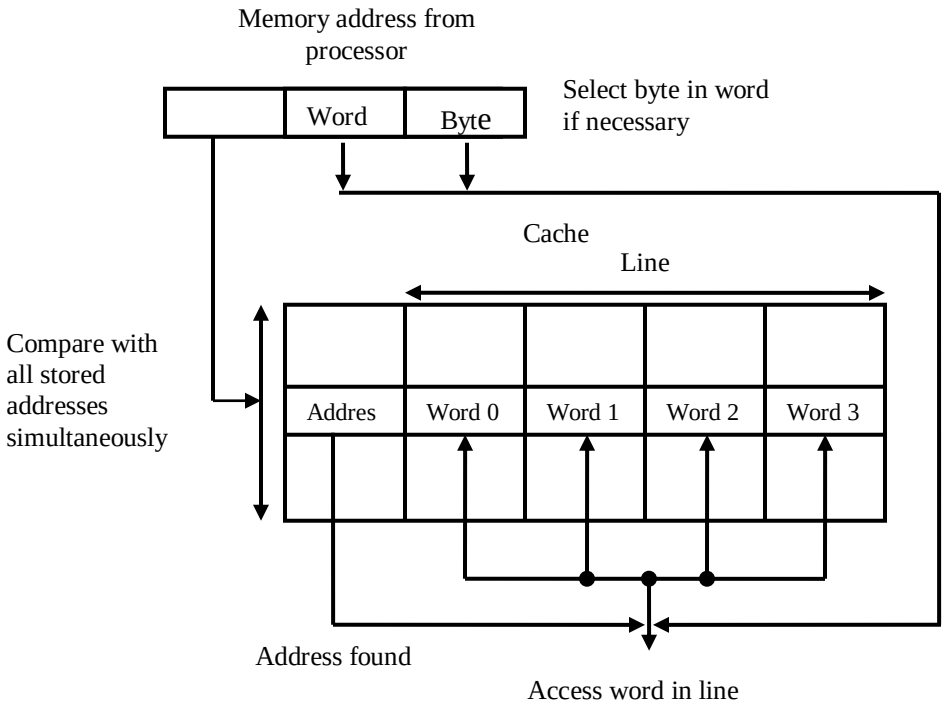
кеш се променя, като оперативната памет може да се промени веднага чрез write through или по-късно (write back).

Както вече споменах информацията в кеш се състои от адрес и данни, като данните могат да се състоят от няколко думи, т. е. блок от последователни адреси или ред, за да има ефект(за да се прояви по-добре) принципа на близостта в пространството.



фиг. 7

В следващия пример (фиг. 8) ред от кеш се състои от четири думи, всяка от които е по четири байта, като целия ред може да бъде трансфериран от оперативната памет в кеш(или обратно) наведнъж ако пътя, по който се придвижват тези данни (data path) е достатъчно голям. Ако размера на този път (data path) е колкото размера на една дума, то редът ще се пренесе на части, като тогава ще е нужно да се съхранява допълнителен бит за всяка дума, който да показва дали в кеша се съдържат валидни данни**.** Минималният размер за ред е 4 байта, но трябва да има възможност да се извърши обръщение към конкретен байт от дадена дума. Когато се извърши обръщение към дума, чрез полето байт се избира байт от тази дума. Най-незначителната част от адреса избира определен байт от дадена дума, следващата част от адреса избира думата, а останалата част се сравнява с адреса в кеш.



фиг. 8

При този метод има възможност блокове от оперативната памет да се заредят в който и да е ред в кеш. Нужен е метод на заместване, чрез който да се избере ред, който да се премахне от кеш при несъответствие, за да може на негово място да се постави новопостъпващия ред от оперативната памет, като алгоритъма трябва да се реализира хардуерно за да се поддържа(запази) високата скорост на изпълнение на операциите(to maintain a high speed of operation). Целта на метода на заместване е да се увеличи коефициента на попадение h.

Основният недостатък на този метод на съответствие е сложността на алгоритъма, т. е. паралелното проверяване на всички адреси в кеш**.** Друг недостатък на метода е, че е скъп за реализиране, поради високата цена на асоциативната памет.

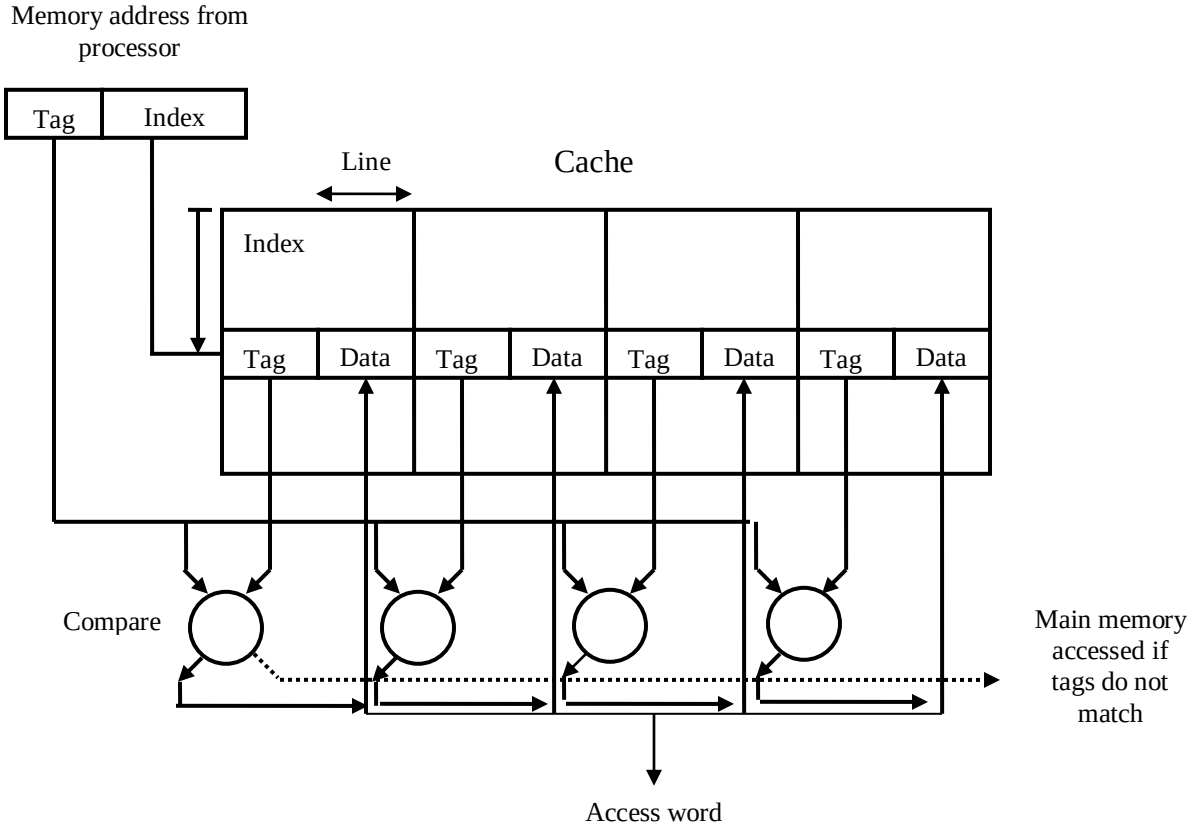
3.Групово-асоциативно съответствие

Групово-асоциативното съответствие позволява ограничен брой редове със същия индекс и с различни етикети да се намират в кеш и затова този метод се счита за компромисен вариант между директното и асоциативното съответствие.

Тук всеки ред в кеш е една група, като групата е разделена на клетки. Броя на редовете в група е известен като асоциативност или размер на групата. Всеки ред във всяка група има етикет, който заедно

с индекса (номера на групата) напълно идентифицира реда. Всяка клетка се състои от етикет и данни. Цялата група е с един и същи индекс, но отделните клетки могат да идват от различни адреси на оперативната памет, т. е. те имат различни етикети. В една група може да се съхранява информация от 4 различни адреса от оперативната памет с един и същи индекс.

Адреса от процесора се разделя на етикет и индекс (фиг. 9). Етикета представлява най-значителните битове от целия адрес. С индекса попадаме в група. Етикета от адреса, който постъпва от процесора го сравняваме едновременно с етикетите на четирите клетки в групата. Ако има съответствие с някои от етикетите, то търсената информация се намира в кеш и се предоставя на процесора, а ако няма – се извършва обръщение към оперативната памет.



фиг. 9

Съответствието между съхранените и идващия етикет от оперативната памет се осъществява чрез използване на компаратори. Броя на компараторите, които за нужни при групово-асоциативното съответствие се определя от броя на редовете в група, а не от броя на

всички редове както е при напълно-асоциативното съответствие. При него всеки етикет изисква свой собствен компаратор. При групово-асоциативното съответствие групата може да бъде избрана бързо и всички редове от групата могат да бъдат прочетени едновременно с етикетите преди да се извършат сравненията на етикетите. След като даден етикет е бил идентифициран, т. е. при попадение, може да се извърши обръщение към съответния ред.

Като заключение можем да кажем, че когато е нужна някаква информация от кеш, кеш контролера не претърсва целия кеш а само реда, чийто адрес съвпада с адреса на търсената информация. Ако търсения ред се намира в кеш се гарантира, че ще бъде в групата, която се претърсва. Така, ако реда не се намира в тази група то той не се намира в кеш и не е нужно кеш контролера да продължава с търсенето.

Методи на заместване

Когато търсената дума не се съдържа в кеша, видяхме, че е необходимо да се извърши трансфер на реда, в който се съдържа тази дума от оперативната памет в кеш, като ако в кеша няма свободно място е необходимо да се извърши заместване, т. е. трябва някой ред от кеш да се освободи за да може да се разположи новопостъпващия ред.

С изключение на директното съответствие, при което не е нужен метод на заместване редът, който трябва да се премахне от кеш се определя чрез метод на заместване. При кеш системите най-използвания метод е метода на най-отдавна неизползвания (least recently used).

Метода на заместване трябва да се реализира изцяло хардуерно, като е за предпочитане избора на ред от кеш, който трябва да се премахне, да се направи по време на извличане на новия ред от оперативната памет. Най-добре ще бъде ако реда, който ще се премахне от кеша няма да е необходим в бъдеще. Но това не може да се предвиди и затова трябва да се вземе решение веднага на базата на известни факти.

При напълно асоциативните кешове при заместване е нужно да се вземат под внимание всички редове, които се намират в кеш. При групово-асоциативните кешове трябва да се вземат под внимание в момента на заместване, само редовете в съответната група, т. е. какъвто и метод на заместване да се използва, при групово-асоциативните кешове трябва да се вземат под внимание много по-малко редове, отколкото при напълно-асоциативните кешове.

При метода “случаен избор” (random replacement algorithm) редът, който трябва да се замести се избира напълно произволно, без да

се имат в предвид предишни обръщения към оперативната памет, затова този метод няма да даде добри резултати, тъй като може да се освободи клетка, която ще бъде необходима в бъдеще.

Друг метод на заместване е first in first out (FIFO) – при този метод, който е влязъл най-напред и най-много е престоял в кеш първи трябва да се освободи. Този алгоритъм може да се реализира с една FIFO опашка, в която се съдържат адресите на редовете, но по-лесен начин за реализиране е като се използват броячи – по един брояч при напълно-асоциативни кешове или по един брояч за всяка група при групово-асоциативни кешове, всеки един с достатъчен брой битове, с които да се идентифицира реда.

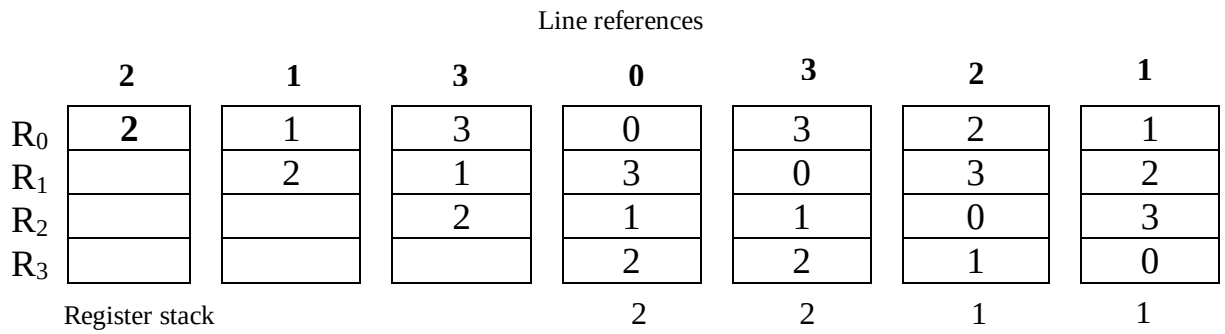
Модификация на този метод е Least recently used(LRU). Идеята на LRU метода е, че колкото по-скоро е използвана дадена информация, толкова е по-голяма вероятността тя да бъде използвана отново, а тази , която е използвана отдавна, може би няма да е необходима вече. Метода е много популярен при кеш системите и може да се реализира лесно, когато броя на редовете е малък.

LRU метода се реализира по много начини: чрез използване на броячи, регистров стек, приблизителни методи и реферационни матрици /reference matrix/. При първия метод всеки ред от кеш е асоцииран с брояч. Един от начините е да се увеличи всеки брояч през определени интервали от време и да се нулира брояча на реда, към който се е извършило обръщение. Така, стойността на всеки брояч ще показва “възрастта” на всеки ред и реда с най-голямата възраст, т. е. реда, който най-дълго е престоял в кеш ще бъде премахнат, когато е нужно да се освободи място в кеш.

Този алгоритъм може да се промени така, че да взима под внимание факта, че броячите имат фиксиран брой битове и да се отчита само относителната възраст. Когато търсената информация се намери в кеша, брояча, асоцииран с реда, в който се съдържа тази информация да се установи в нула. Това показва, че този ред е най-скоро използвания и всички броячи, които имат по-малка стойност от стойността на брояча на реда, в който се съдържа търсената информация, се увеличават с единица, а всички броячи, които имат по-голяма стойност не се пипат, тъй като те не оказват никакво влияние. Когато търсената информация не се намери в кеша, т. е. при непопадение при положение, че кеша не е пълен, брояча асоцииран с новопостъпващата информация в кеш, се установява в нула и всички други броячи се увеличават с единица. В случай на непопадение, при положение че кеша е пълен, реда, чийто брояч е с максимална стойност

се премахва от кеш, за да се разположи на негово място новопостъпващия ред, след което брояча се установява в нула, а всички други броячи се увеличават с единица. Брояча с най-голямата стойност идентифицира най-отдавна неизползвания ред.

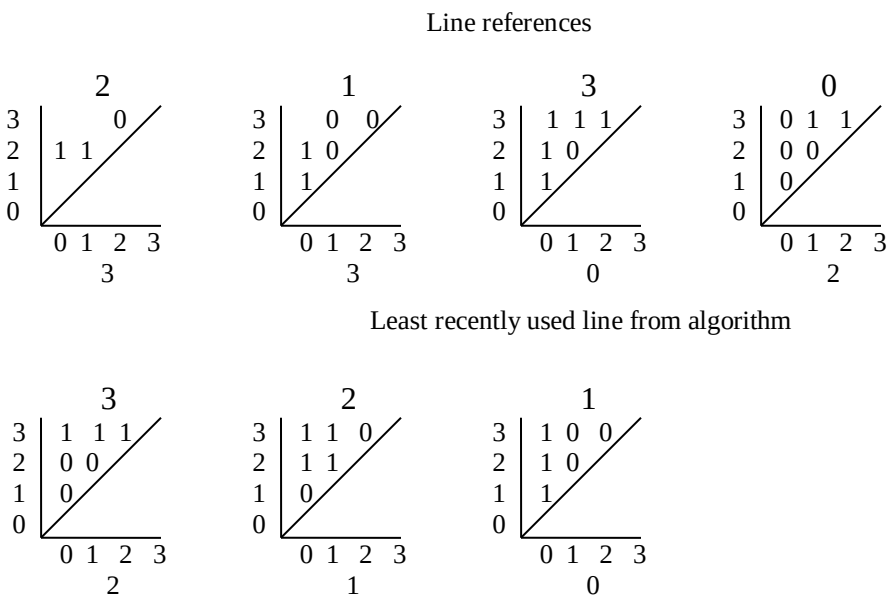
При използване на регистров стек се прави следното: формират се n-битови регистри, по един за всеки ред в група (фиг. ?? – *защо са въпросите*).



Най-скоро използвания ред се поставя най-отгоре в стека, а най-отдавна неизползвания – съответно на дъното на стека. Когато се извърши обръщение към даден ред, този ред се поставя на най-първата позиция в стека, като всички останали стойности, които се намират в стека се изместват с една позиция надолу, до момента в който ще се установи, че в регистъра се съдържа същата стойност като новопостъпващата.

Друг начин на реализиране на LRU метода е чрез реферационни матрици.

При този метод вниманието се съсредоточава върху матрица от битове. Има няколо версии на този метод. В една от тях горния триъгълник в матрица с размер BxB се формира без диагонала, при положение, че има B на брой реда, които трябва да се разгледат. Този триъгълник съдържа (Bx(B - 1))/2 бита. Когато се извърши обръщение към i – тия ред всички битове от i – тия ред на матрицата се установяват в единица, след което всички битове в i – тата колона се установяват в нула. Най-отдавна неизползвания ред е този, който съдържа само нули в неговия ред и само единици в неговата колона, като той лесно може да се открие. Метода е показан на фиг. 3.10 при B = 4, като последователността от обръщения е към редовете 2, 1, 3, 0, 3, 2, 1, ..., заедно със стойностите, които ще се получат използвайки регистров стек.



Друг метод на заместване е Least-frequently used (метод на най-малко използвания). При него се замества този ред от кеш, към който са извършени най-малко обръщения.

1. Влияние на методите на съответствие при различни размери на кеша

Целта е да се анализира влиянието на метода на съответствие върху степента на несъвпадение при различни размери на кеша.

Задачи:

Конфигурирайте система със следните архитектурни характеристики:

- Брой процесори = 1;
- Протокол за кохерентност на кеша = MESI;
- Scheme for bus arbitration = Random;
- Дължина на думата = 32;
- Брой думи в блок = 64 (размер на блока = 256 байта);
- Брой блокове в оперативната памет = 4096 (размер на оперативната памет = 1024 KB);
- Метод на заместване = LRU (метод на най-отдавна неизползвания);

Конфигурирайте метода на съответствие използвайки следните конфигурации: директно, групово-асоциативно с два реда в група (2-way), групово-асоциативно с 4 реда в група (4-way), групово-асоциативно с 8 реда в група (8-way) и напълно-асоциативно

съответствие (запомнете: броя на редовете в група се определя като броя на блоковете в кеш се раздели на броя на групите). За всяка една от тези конфигурации конфигурирайте броя на блоковете в кеш за да получите следните размери за кеш паметта: 4KB (16 блока в кеш), 8 KB, 16 KB и 32 KB (128 блока в кеш). За всяка конфигурация получете степента на несъвпадение използвайки трейса *Ear*.

Въпроси:

Степента на несъвпадение увеличава ли се или намалява с увеличаване на асоциативността? Защо? Какво се случва с conflict misses с увеличаване на асоциативността?

Влиянието на асоциативността увеличава ли се или намалява с увеличаване размера на кеша? Защо?

В заключение, увеличаването на асоциативността подобрява ли производителността на системата? Ако отговорът е да, при кой преход се получават най-голямо подобрение на производителността: от директно към групово-асоциативно с два реда в група, от групово-асоциативно с два реда в група към групово-асоциативно с четири реда в група, от групово-асоциативно с четири реда в група към групово-асоциативно с осем реда в група или от групово-асоциативно с осем реда в група към напълно-асоциативно?

2. Влияние на методите на заместване

Целта е да се покаже влиянието на метода на заместване върху степента на несъвпадение.

Задачи:

Конфигурирайте система със следните архитектурни характеритики:

- Брой процесори = 1;
- Протокол за кохерентност на кеша = MESI;
- Scheme for bus arbitration = Random;
- Дължина на думата (в битове) = 16;
- Брой думи в блок = 16 (размер на блока = 32 байта);
- Брой блокове в оперативната памет = 8192 (размер на оперативната памет = 256 KB);
- Брой блокове в кеш = 128 (размер на кеш паметта = 4 KB);
- Метод на съответствие = Групово-асоциативно съответствие с осем реда в група (брой на групите в кеш = 16);

Конфигурирайте метода на заместване използвайки следните конфигурации: Random, LRU, LFU и FIFO. За всяка от тези конфигурации получете степента на несъвпадение използвайки трейс файловете (разширение ".prg"): *Hydro, Nasa7, Cexp, Mdljd, Ear, Comp,Wave, Swm и UComp.*

Въпроси:

При кой метод на заместване се получава най-малка степен на несъвпадение? А при кой се получава най-голяма степен на несъвпадение? Do the benefits of LFU and FIFO policies happen for all the benchmarks or do they depend on the different locality grades?

Мислите ли че при различните методи на заместване резултатите ще бъдат различни ако се използва директно съответствие? Защо?

В заключение, правилния избор на метод на заместване подобрява ли производителността на системата? Ако отговорът е да, кой е преходът при който се получават най-добри резултати: от Random към LRU, от Random към LFU или от Random към FIFO? Защо (consider the cost/performance aspect)?

3.Проявление на принципа на близостта при различните програми (Program locality)

Целта е да се покаже, че при различните програми принципа на близостта се проявява по различен начин, като при едни програми се изразява по-добре отколкото при други.

Задачи:

Конфигурирайте система със следните архитектурни характеристики:

- Брой процесори = 1;
- Протокол за кохерентност на кеша = MESI;
- Scheme for bus arbitration = Random;
- Дължина на думата = 16;
- Брой думи в блок = 16 (размер на блока = 32 байта);
- Брой блокове в оперативната памет = 8192 (размер на оперативната памет = 256 KB);
- Брой блокове в кеш = 128 (размер на кеш паметта = 4 KB);
- Метод на съответствие = Напълно асоциативно съответствие;
- Метод на заместване = LRU (метод на най-отдавна неизползвания);

Получете степента на несъвпадение използвайки следните трейсове: *Hydro, Nasa7, Cexp, Mdljd, Ear, Comp, Wave, Swm и Ucomp*(трейс файлове със същото име и разширение ".prg").

Въпроси:

При всички програми ли принципа на близостта се проявява по един и същи начин? Коя е програмата, при която най-добре се проявява този принцип? А коя е с the worst? Do you think that the design of memory systems that exploit the locality of certain kind of programs (which will be the most common in a system) can increase the system performance? Why?

По време на експеримента може да се наблюдава графично как всъщност степента на несъвпадение намалява по време на изпълнение на програмата. Защо? Каква е причината?

Упражнение 1

Разучаване на симулатора SMP CACHE

1. Въведение

Това е симулатор за анализиране и изучаване на кеш памети при многопроцесорните системи. Този симулатор е с графичен интерфейс, който е лесен за разбиране (използване) и работи на компютърни системи с операционна система Windows.. Някои от параметрите, които могат да се изучат с този симулатор са: program locality, влиянието на броя на процесорите, протоколите за кохерентност на кеша, schemes for bus arbitration, методите на съответствие, методите на заместване, размера на кеша (броя на блоковете в кеш), броя на групите в кеш (при групово-асоциативните кешове), броя на думите в блок (размера на блока в оперативната памет), дължина на думата и др.

Заради лесния и user friendly интерфейс този симулатор се препоръчва за учебни цели. Той ни позволява да наблюдаваме развитието на многопроцесорната система, по време на изпълнение на дадена програма.

2. Configuration files

Симулаторът ни позволява да си направим собствена конфигурация(т. е. ние да определим какви да бъдат стойностите за различните характеристики), която можем да съхраним във файл с разширение ".cfg", за да може по-късно пак да я използваме.

Всички характеристики си имат кодове, които са показани в следващата таблица:

	Possible Values	Code
Cache coherence protocol	MSI MESI DRAGON	1 2 3
Scheme for bus arbitration	Random LRU LFU	1 2 3
Mapping	Direct Set-associative Fully-associative	1 2 3
Cache sets	NO 1, 2, 4, …, 2048	0 1, 2, 4, …, 2048
Replacement policy	NO Random	0 1

	LRU FIFO LFU	2 3 4
Writing strategy	Write-through Write-back	1 2

Въпреки, че има само по една възможност за методите на запис (Write-back) и за нивата на кешовете (Level 1), те трябва да присъстват в configuration files.

3. Trace files

За да можем да работим със симулатора е необходимо да използваме файлове с разширение ".prg", т. е. файлове с "calls"и адресите от оперативната памет, към които ще се обръщат процесорите по време на изпълнение на дадена програма. Това са така наречените трейсове. Тези трейс файлове се състоят от редове, всеки от които има два номера: етикет (label) и стойност(value). Етикетът представлява десетичен номер, който идентифицира типа на операцията за обръщение на процесора към адрес от оперативната памет в даден момент according to the instruction program: за извличане на инструкция (0), за за четене на данни от оперативната памет (2), за запис на данни в оперативната памет (3). Стойността представлява номер, който показва адреса на думата от оперативната памет, към която ще се обърне процесора. This address will be translated by the simulator for locating the word in the memory system block structure.

На следващата фигура е показан пример за трейс файл, който може да се зареди в някой от процесорите на нашия SMP.

```
0    00001b08
0    00001ca5
2    00007951
0    00001d04
0    00001eb8
2    00007952
0    0000201c
2    00007c71
0    0000201f
3    00007b51
```

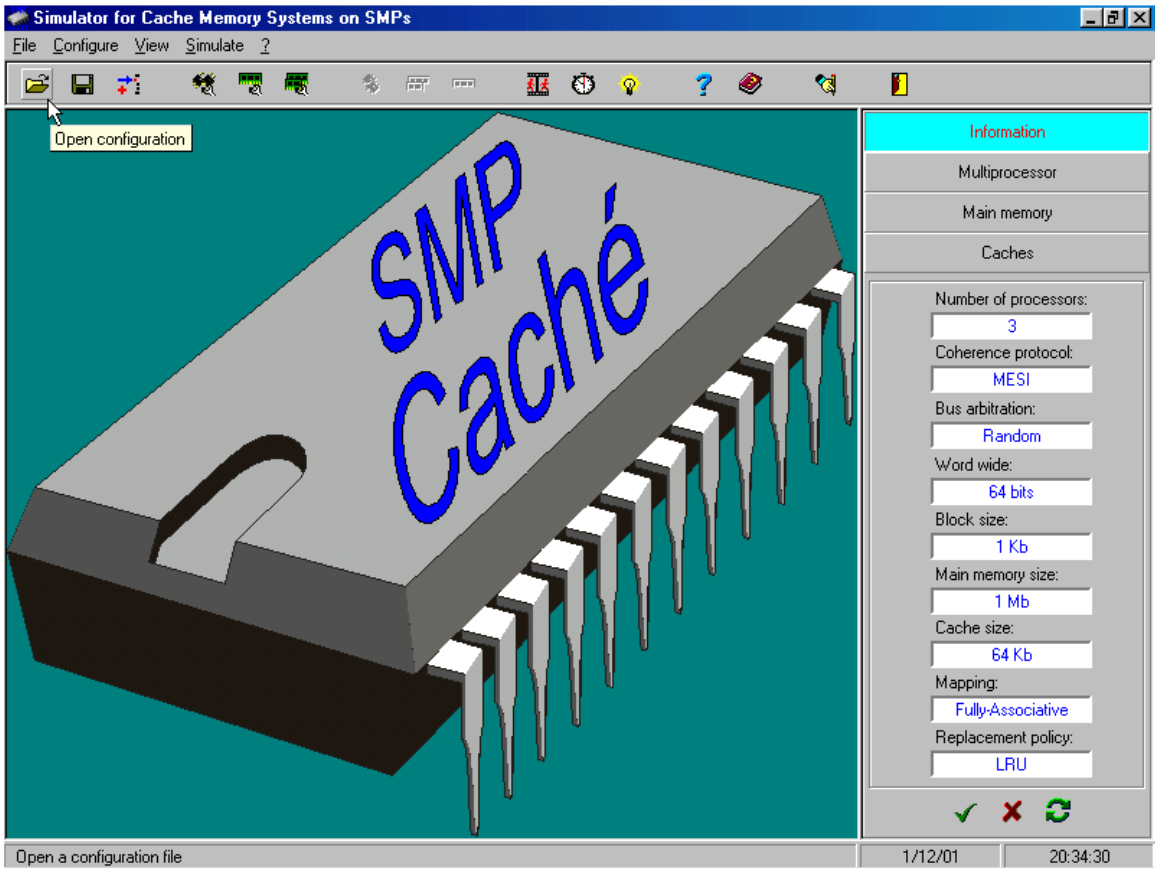
фиг. Примерен трейс файл

Имаме 6 обръщения към паметта за извличане на инструкция, 3 обръщения за четене на информация от оперативната памет и едно

обръщение за запис. Общо са необходими десет обръщения към оперативната памет.

4. Interface overview

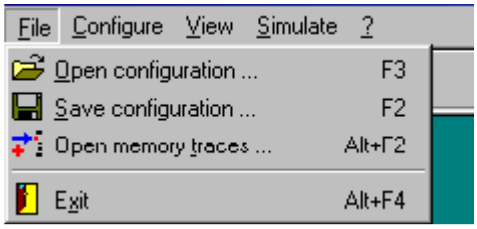
Кликайки върху иконата на SMP Cache първото нещо, което виждаме е следното:



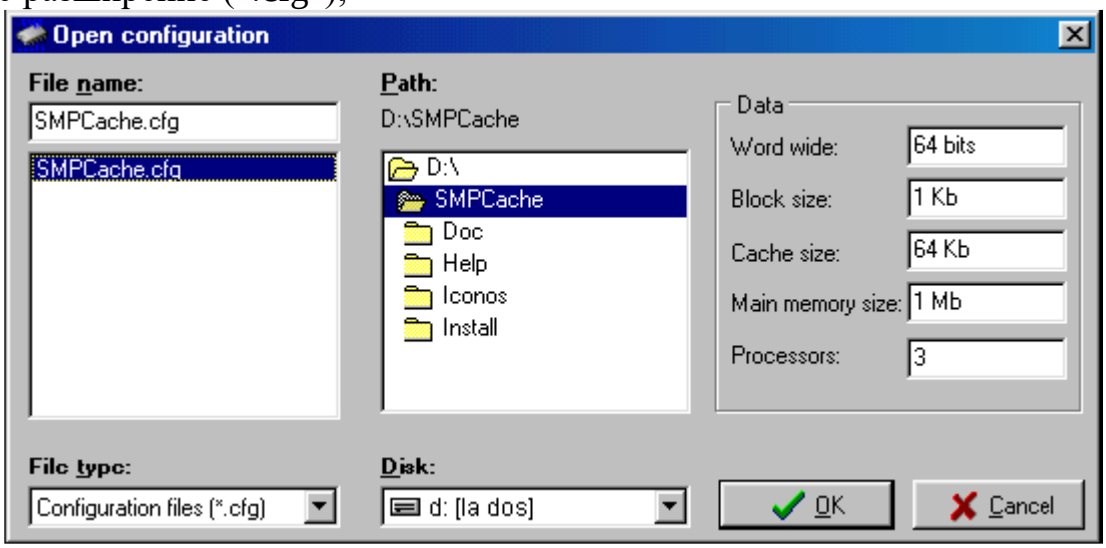
Този екран съдържа лента с менюта, лента с инструменти, configuration panel и статус бар.

4.1. Menu bar – разположено е над лентата с инструменти. То съдържа падащи менюта, съдържащи различни команди. Лентата с менюта съдържа следните менюта: File, Configure, View, Simulate и Help.

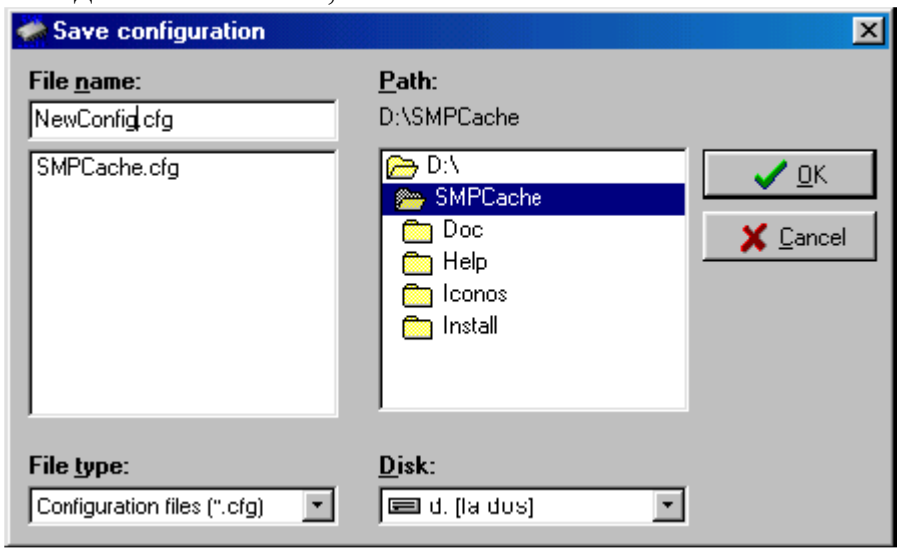
4.1.1. File menu – То съдържа следните команди: Open Configuration, Save configuration, Open memory traces и Exit.



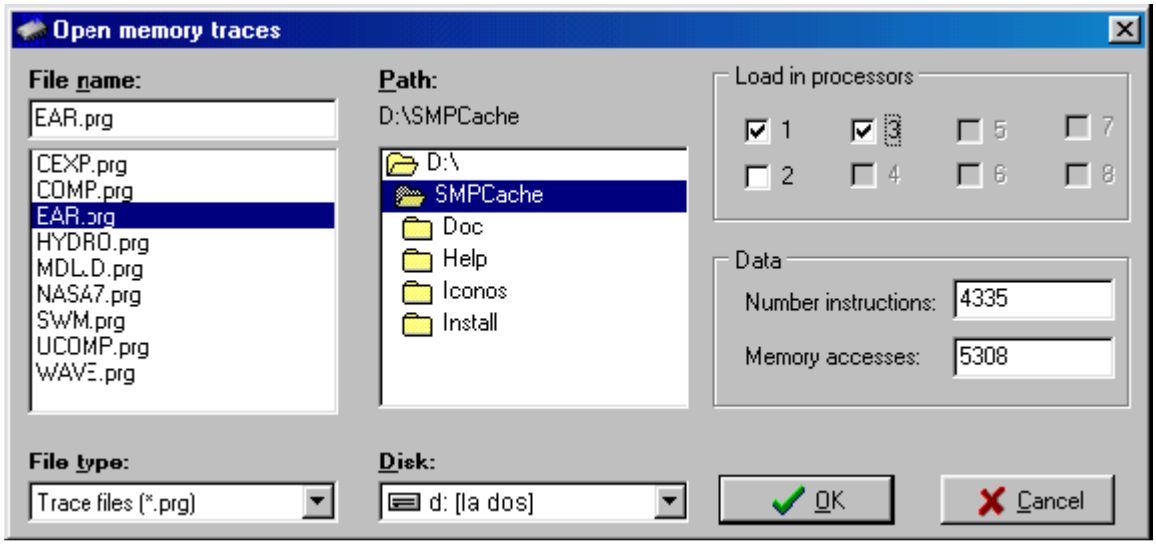
А) Open Configuration – тази команда ни позволява да отворим файл с разширение (“.cfg”);



Б) Save configuration – тази команда ни позволява да съхраним във файл с разширение (“.cfg”) текущата конфигурация за да може по-късно пак да я използваме;

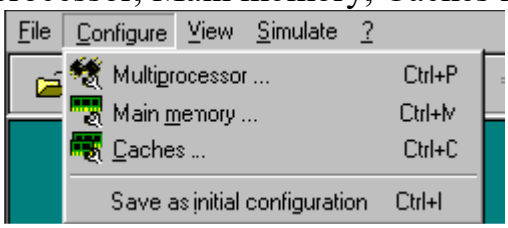


В) Open memory traces – тази команда ни позволява да отворим файл с разширение (“.prg”), който трябва да заредим(зареждайки го) в някои от активните процесори;

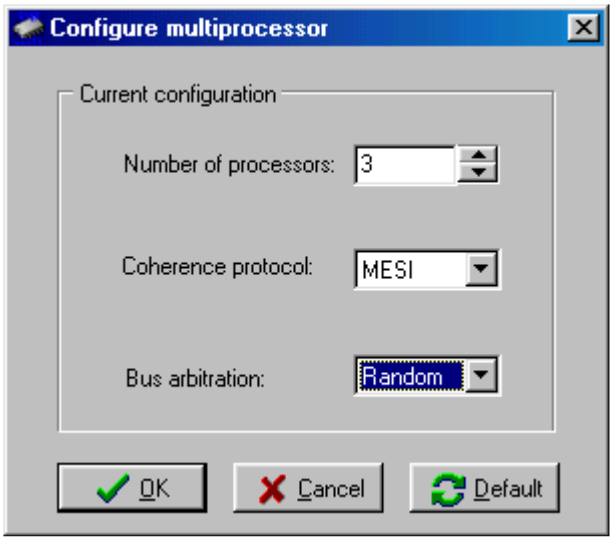


Г) Exit – чрез тази команда излизаме от SMP Cache;

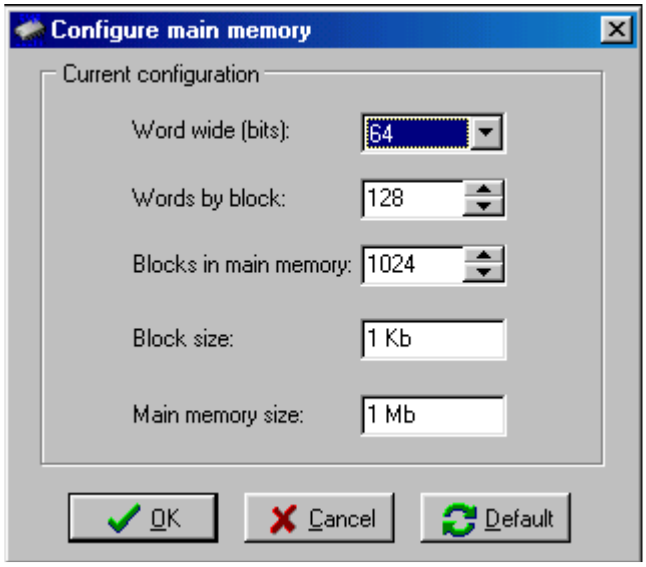
4.1.2. Configure menu – то съдържа следните команди: Multiprocessor, Main memory, Caches и Save as initial configuration.



А) Multiprocessor – тази команда ни позволява да си направим нова конфигурация включваща брой процесори, протокол за кохерентност и bus arbitration.

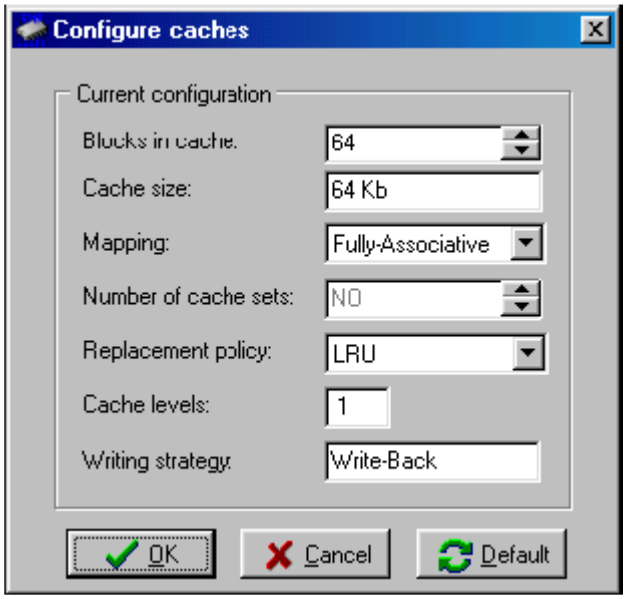


Б) Main Memory – тази команда ни позволява да си направим нова конфигурация на паметта.



Параметрите, за които трябва да зададем стойности са дължина на думата, брой думи в блок, брой блокове в оперативната памет, размер на блока и размер на оперативната памет.

В) Caches – тази команда ни позволява да си направим конфигурация на кешовете свързани с всеки процесор. Всички кешове ще имат същата конфигурация. Параметрите вързани с кешовете са брой блокове, размер на кеша, метод на съответствие, брой групи в кеш (при групово-асоциативните кешове), метод на заместване, ниво на кеша, метод за запис.



Максималния размер за кеша е 16MB (защото максималния размер за блока е 8KB).

Г) Save as initial configuration – тази команда ни позволява да съхраним текущата конфигурация като конфигурация по подразбиране.

4.1.3. Simulate menu – то съдържа следните команди: Step by step, With breakpoint и Complete execution, т. е. това меню съдържа командите свързани с начина на изпълнение на симулациите.

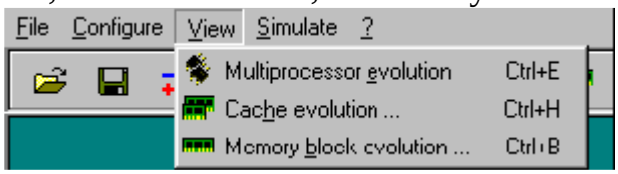
Има три начина:

1) Step by step – ако изберем тази команда симулацията се спира след всяко обръщение към оперативната памет. Този вид симулация е по подразбиране;

2) With breakpoint – ако изберем тази команда симулацията се спира след извършване на определен брой обръщения към оперативната памет;

3) Complete execution – ако изберем тази команда симулацията се спира след извършване на всички обръщения към паметта (when all the memory traces are finished);

4.1.4. View Menu – то съдържа следните команди: Multiprocessor evolution, Cache evolution, и Memory block evolution.



За да можем да изберем дадена команда от това меню е необходимо да сме заредили поне един трейс файл в някой процесор.

А) Multiprocessor evolution – тази команда ни позволява да имаме глобален поглед върху развитието на многопроцесорната система в зависимост от направената конфигурация. Симулаторът ни показва как реагира системата(responds) на обръщенията към оперативната памет генерирани от програмите (трейс файловете, използвани за различните процесори по време на симулацията).

Само процесорите, в които е зареден трейс влияят на симулацията.

При избирането на командата Multiprocessor evolution се отваря прозорец, на който е показана диаграма на SMP. На тази диаграма са показани bus transactions (BusRd, BusRdX, BusWB and BusUpd), изканията на процесора (PrRd XXX and PrWr XXX) и трансфера на

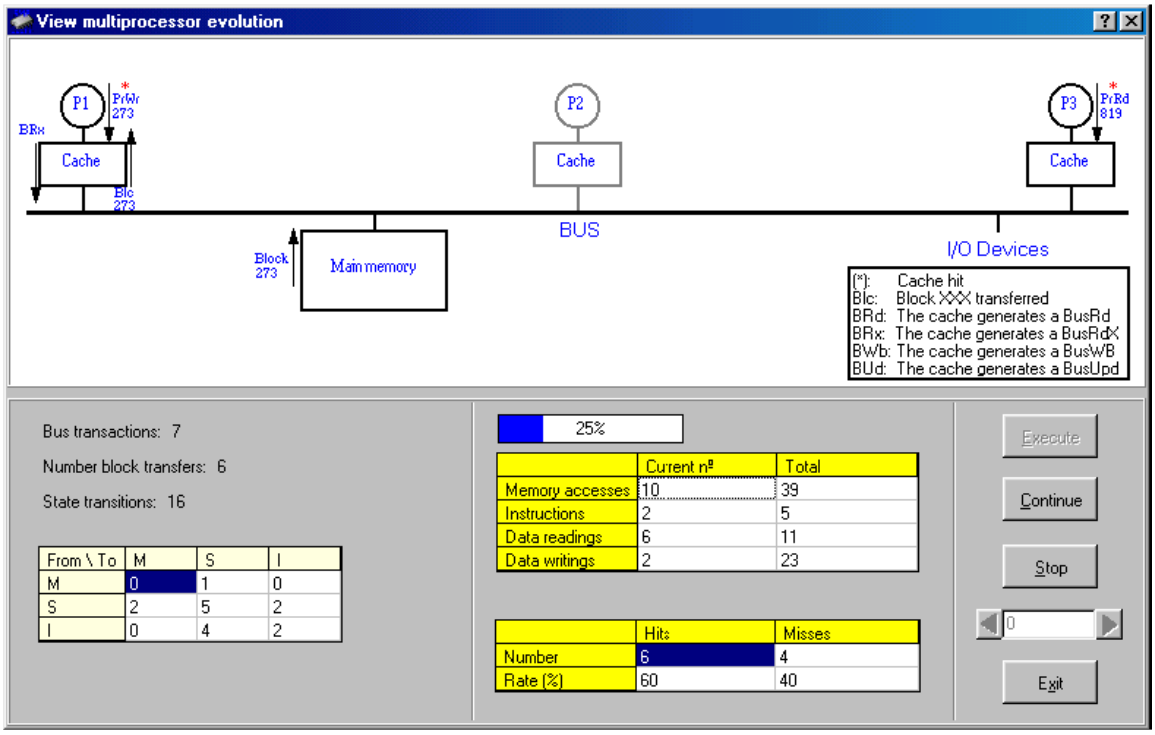
блоковете (Block XXX). Със стрелка се показва посоката на bus transaction, исканията на процесора или трансфера на блока.

Bus transactions винаги са от даден кеш към шината.

Когато процесорът има някакво искане той се обръща към неговия кеш. Да предположим, че процесорът иска две неща: четене (PrRd) и запис (PrWr). Извличането на инструкции и четенето на данни се считат за PrRd искане. Четене или запис може да се извършва в блок от оперативната памет,като не е задължително този блок да се намира и в кеш.

Когато се извърши трансфер на даден блок (блок XXX), XXX е адреса на този блок в оперативната памет.

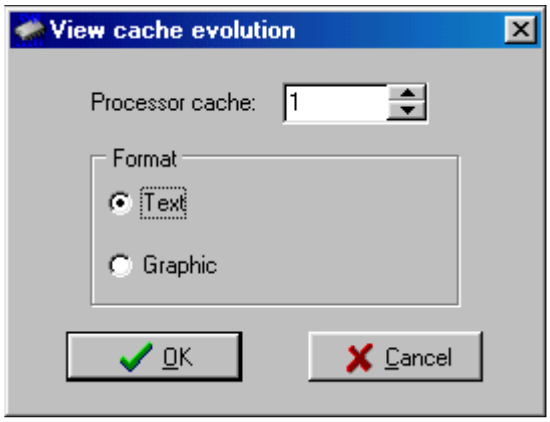
На този прозорец още се показват броя на bus transactions, броя на блоковете трансферирани по шината, общия брой обръщения към паметта, типа на обръщение (дали е за извличане на инструкция, за четене на данни или за запис на данни), броя на попаденията и несъвпаденията (степента на попадения и степента на несъвпадение) и др.



За да стартирате симулацията кликнете върху бутона Execute. Може да прекъснете симулацията по всяко време кликайки върху бутона Exit. Бутонът Stop ни позволява да спрем незабавно дадена

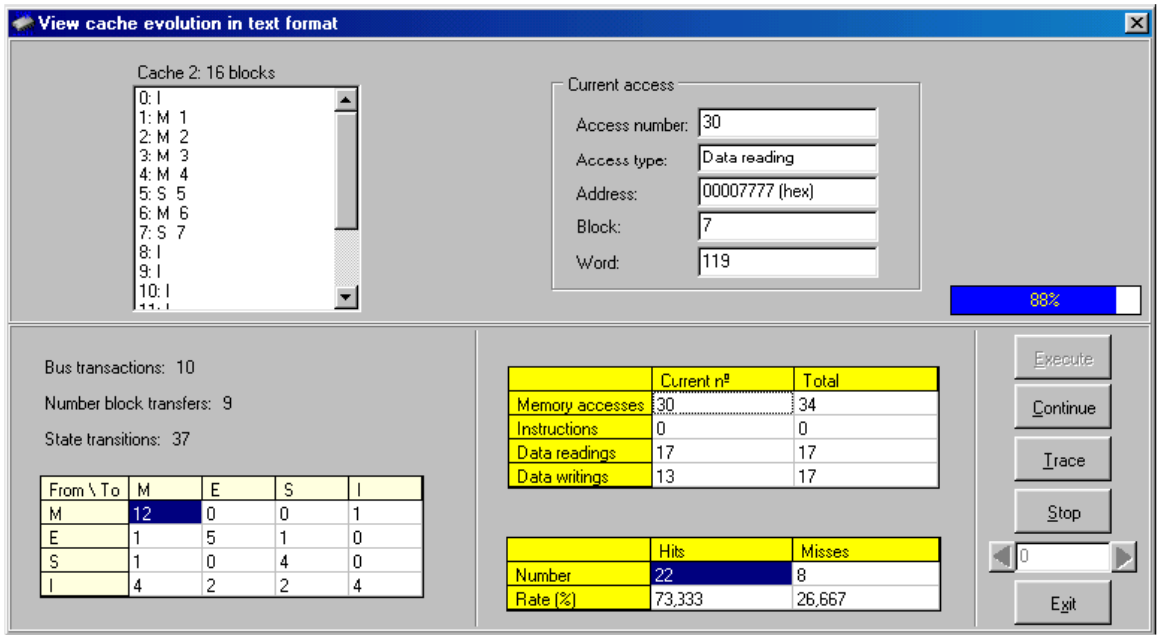
симулация. За да продължите същата симулация кликнете върху бутона Continue.

Б) Cache Evolution – Тази команда ви позволява да наблюдавате развитието на конкретен кеш според направената конфигурация и обръщенията към паметта, които се генерират от програмите(трейс файловете). На фигурата е показана диалогова кутия, която се отваря избирайки тази команда.



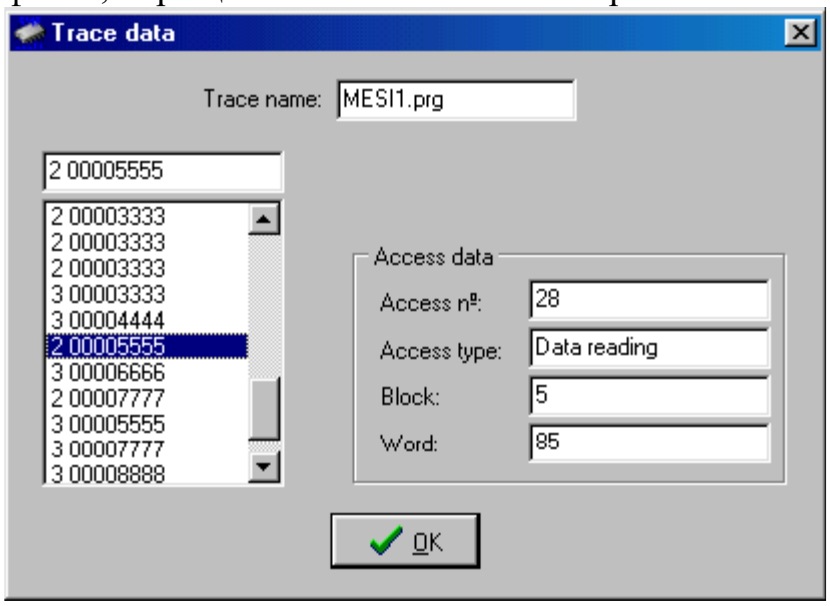
Можете да изберете кеша, който искате да изследвате и начина, по който да бъдат показани резултатите(текстово или графично), след което трябва да кликнете върху бутона OK.

Ако изберете текстов формат се отваря следния прозорец:



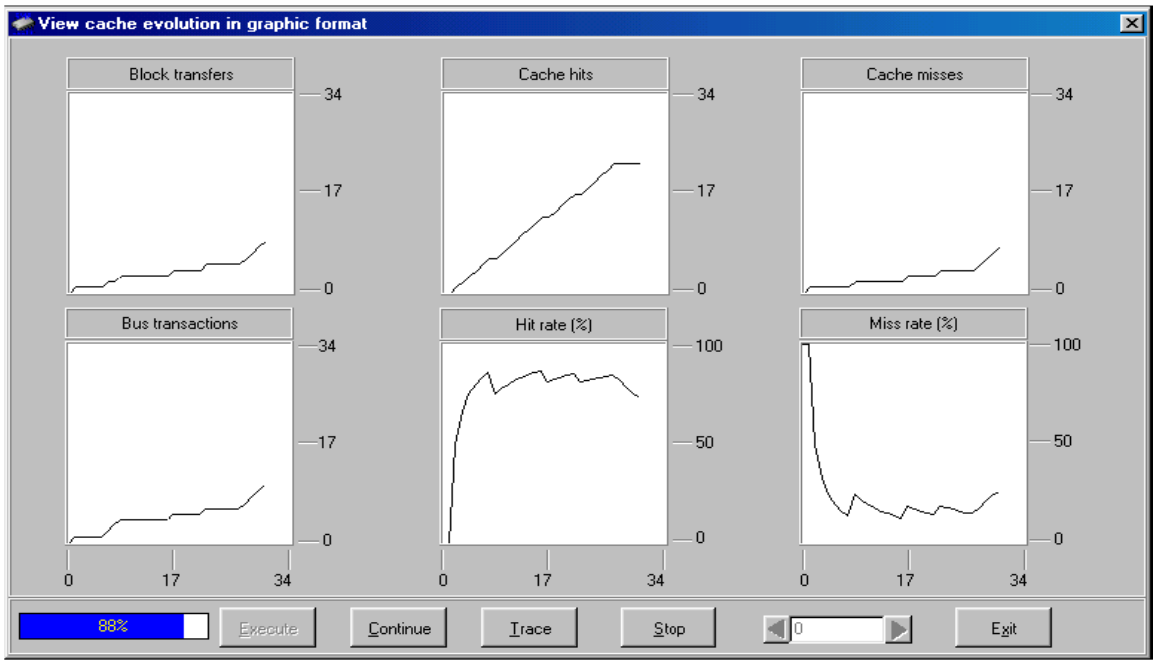
На този прозорец са показани номера на кеша(процесора) броя на блоковете в кеш, състоянието на всеки блок от кеш и др.

От тази диалогова кутия можете да получите подробна информация за трейса, асоцииран с този кеш (процесор): име на трейс файла, обръщенията към паметта по време на изпълнение на този трейс



фиг. Trace data

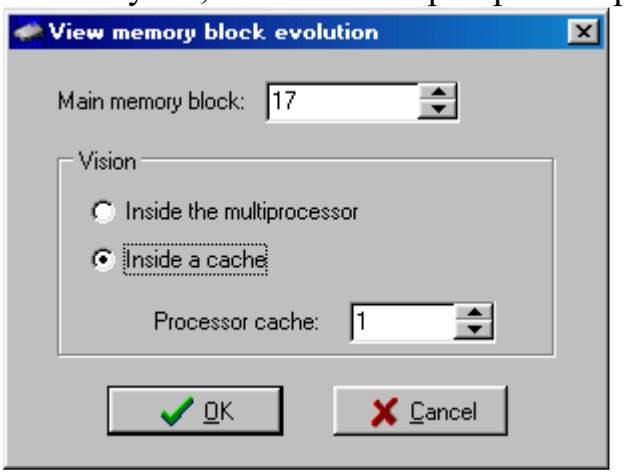
Ако изберете графичен формат се отваря прозореца, показан на следващата фигура:



View cache evolution in graphic format

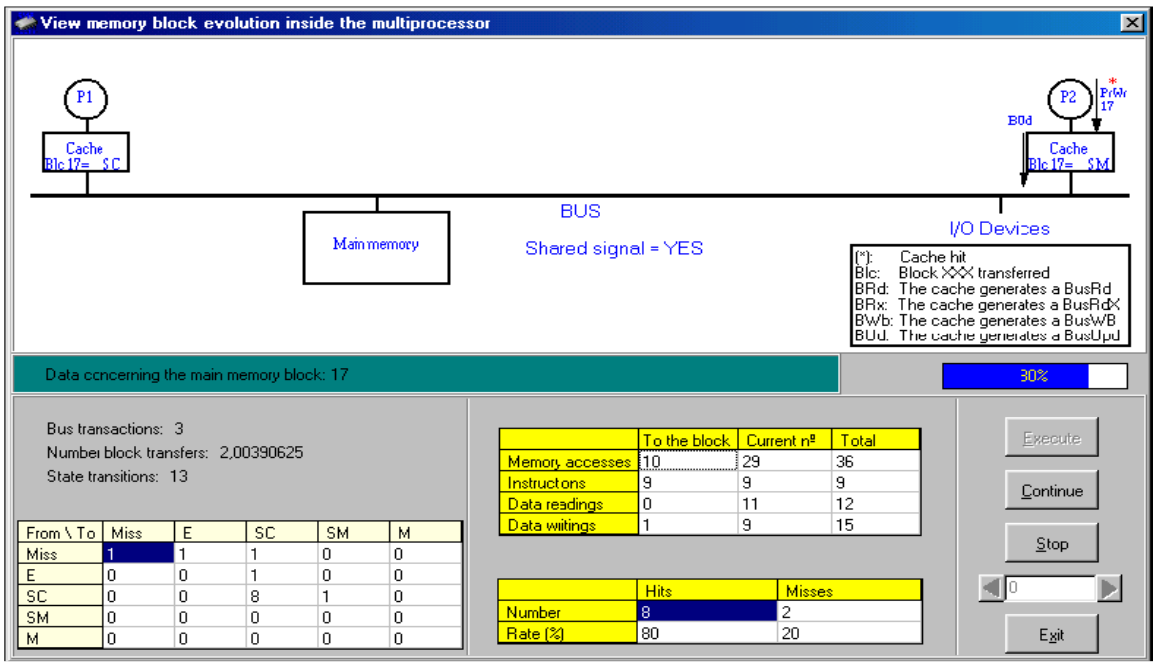
Тук са показани броя на трансферираните блокове по шината, броя на транзакциите по шината генерирани от този кеш, броя на попаденията и несъвпаденията, както и степента на попадение и степента на несъвпадение.

Memory block evolution – тази команда ви позволява да наблюдавате конкретен блок от конкретен кеш. На фигурата е показана диалоговата кутия, която се отваря при избор на тази команда.

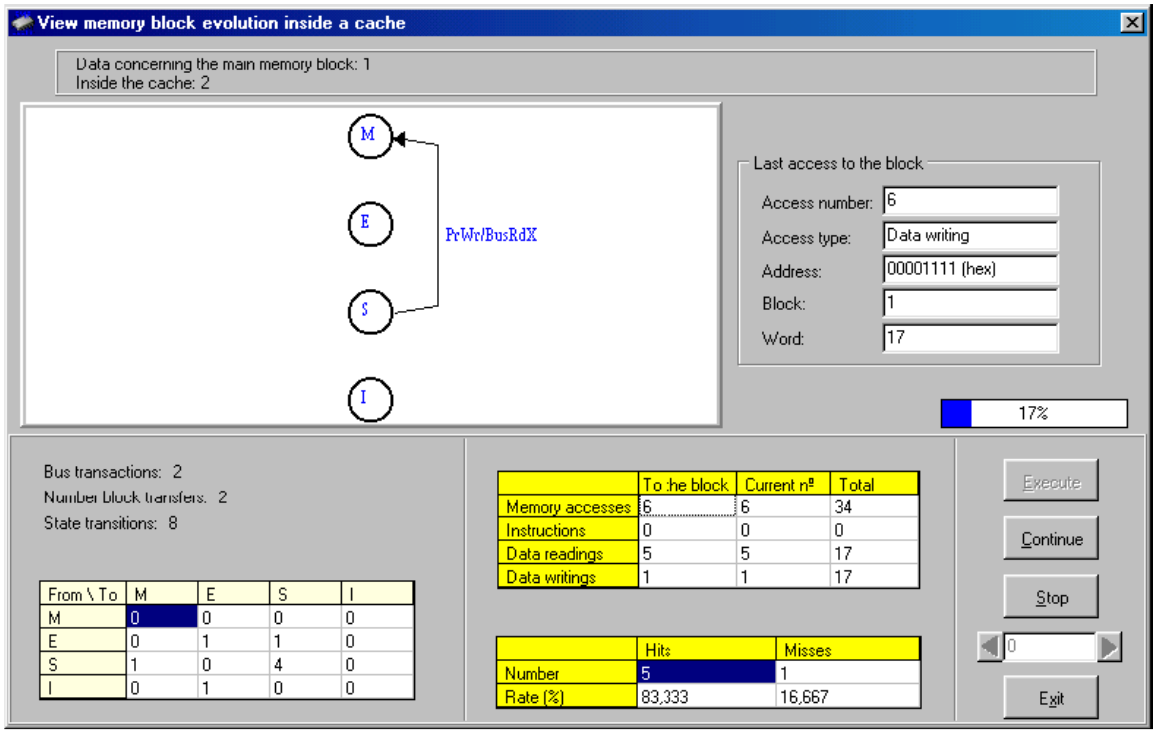


Можете да изберете блока от оперативната памет, който искате да изследвате and the desired vision level: inside the multiprocessor or inside a specific cache (може да бъде кеш, в чийто процесор е зареден трейс). След това кликнете върху бутона OK.

Ако изберете изглед на многопроцесорната система ще се отвори следния прозорец:



Ако изберете поглед върху кеш паметта се отваря следния прозорец:



За да стартирате симулацията кликнете върху бутона Execute. Може да прекъснете симулацията по всяко време кликайки върху бутона Exit. Бутонът Stop ви позволява да спрем незабавно дадена симулация. За да продължите същата симулация кликнете върху бутона Continue.

5.Configuration Panel

Този панел ни позволява да задаваме различни стойности за различните параметри за да получим конкретна архитектура. Съдържа следните ..... Information, Multiprocessor, Main memory, and Caches. Information показва информация за текущата конфигурация, а другите са подобни на командите, които се съдържат в менюто Configure.

6.Задачи за упражнение:

1. Задайте стойности за различните параметри:

А) използвайки the default initial configuration of the simulator;

Б) променяйки конфигурацията по подразбиране, като използвате менюто Configure или Configuration panel;

В) зареждайки вече съхранена конфигурация;

2.Заредете трейсове в активните процесори, използвайки командата “Open memory traces”. Помнете, че трябва да използвате тази команда за всеки различен трейс.

3. Изберете начин на изпълнение на симулацията, като използвате менюто Simulate (Step by step, With breakpoint и Complete execution) като е възможно да преминете от един начин на изпълнение към друг без да е необходимо да чакате симулацията да приключи.

4. Изберете the vision level и стартирайте симулацията, като използвате менюто View. Може да наблюдавате развитието на всички процесори (Multiprocessor evolution) и всички блокове от оперативната памет, избирайки командата Memory block evolution. Освен това можете да наблюдавате специфичен кеш, избирайки командата Cache evolution и всички блокове от оперативната памет, избирайки командата Memory block evolution.