Лекція

з дисципліни: «Комп’ютерна схемотехніка та архітектура комп’ютерів»

на тему: «Ієрархія пам’яті»

ПЛАН ЛЕКЦІЇ

1. Функція пам’яті.
2. Кеш-пам’ять.
3. Віртуальна пам'ять

ЛІТЕРАТУРА

1. Девід М. Харріс і Сара Л. Харріс. Цифрова схемотехніка та архітектура комп'ютера / пер. з англ. 2-е вид. - Morgan Kaufman, 2013. - 1621 с.
2. Мельник А.О. Персональні суперкомп’ютери: архітектура, проектування, застосування: монографія / А.О. Мельник, В.А. Мельник. – Львів: Видавництво Львівської політехніки, 2013. – 516 с.
3. Матвієнко М. П. Архітектура комп’ютера. Навчальний посібник. / Матвієнко М. П., Розен В. П., Закладний О. М.— К: Видавництво Ліра-К, 2016. — 264 с.

1. Функція пам’яті

Пам’яттю комп’ютера називають сукупність різних пристроїв, призначених для приймання, зберігання і видачі двійкової інформації.

Окремий пристрій називають запам’ятовуючим пристроєм (ЗП) або просто пам’яттю. Термін “запам’ятовуючий пристрій” вживають тоді, коли треба підкреслити принцип його побудови: на магнітному осерді, напівпровідниках тощо. Термін “пам’ять” застосовують, коли вказують на функцію, яку вона виконує: основну, постійну тощо.

Пам’ять комп’ютера функціонує під керуванням операційної системи, яка розміщує масиви інформації в пам’яті, забезпечує їхній захист від несанкціонованого доступу та виконує інші функції. Продуктивність і обчислювальні можливості комп’ютера значною мірою визначаються складом і характеристиками ЗП, які застосовуються.

Класифікація пам’яті

Пам’ять сучасних комп’ютерів класифікують за функціональним призначенням, видом носія інформації, способом організації доступу до інформації. За функціональним призначенням пам’ять комп’ютерів поділяється на дві основні групи: зовнішню і внутрішню.

Зовнішні ЗП призначені для тривалого зберігання великих масивів інформації з ємністю до гігабайта і більше та малою швидкодією. Зовнішня пам’ять містить в собі накопичувачі на магнітних стрічках, дисках, барабанах та оптичних дисках.

Внутрішні ЗП призначені для зберігання програм і даних, які виконуються в поточний момент часу.

Продуктивність комп'ютерної системи залежить від її підсистеми пам'яті так само сильно, як і від мікроархітектури процесора.

Процесор працює з пам'яттю через інтерфейс пам'яті (memory interface). На Рис. 1 показано простий інтерфейс до пам'яті, використаний в нашому багатотактного MIPS процесорі. Процесор виставляє адреси на шину адреси (address bus), що йде до підсистеми пам'яті. Для читання керуючий сигнал записи (MemWrite) встановлюється в 0, а пам'ять повертає дані по шині читання даних (ReadData). Для запису MemWrite встановлюється в 1 і процесор посилає дані в пам'ять по шині записи даних (WriteData).

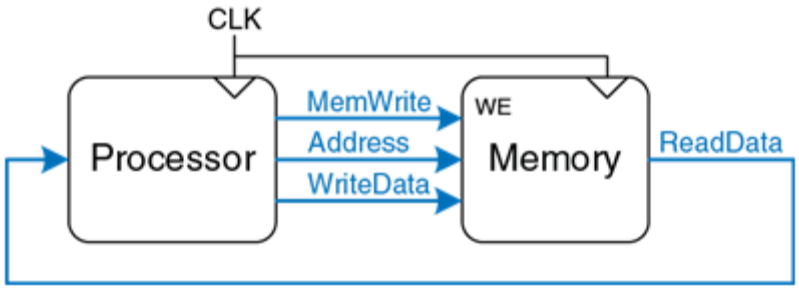


Рис. 1

Підсистеми пам'яті використовують ієрархію сховищ для швидкого доступу до найбільш часто використовуваних даних, одночасно забезпечуючи можливість зберігання великих обсягів даних.

Комп'ютерна пам'ять в основному побудована на базі динамічної (DRAM) і статичної (SRAM) пам'яті. В ідеалі пам'ять повинна бути швидкою, великою і дешевою. Однак на практиці будь-який тип пам'яті має тільки дві з цих властивостей; пам'ять або повільна, або дорога, або маленького обсягу. Незважаючи на це, комп'ютерні системи можуть наближатися до цього ідеалу, поєднуючи дешеву, швидку і маленьку пам'ять з дешевою, повільною і великою. Швидка пам'ять використовується для зберігання часто використовуваних даних і команд, так що створюється враження, що підсистема пам'яті завжди працює досить швидко. Інші дані і команди зберігаються у великій пам'яті, яка працює повільніше, але дозволяє мати великий загальний обсяг пам'яті. Комбінація двох дешевих типів пам'яті - це набагато дешевший варіант, ніж одна велика і швидка пам'ять. Цей принцип поширюється на всю ієрархію пам'яті, так як зі збільшенням обсягу пам'яті зменшується її швидкість роботи.

Оперативна пам'ять комп'ютера зазвичай будується на мікросхемах динамічної пам'яті (DRAM). Проте час доступу до даних в DRAM на порядок або два більший, ніж тривалість такту процесора (десятки наносекунд проти частин наносекунди).

Щоб впоратися з цією проблемою, комп'ютери зберігають найбільш часто використовувані команди і дані в швидкій, але невеликий за обсягом пам'яті, званої кеш-пам'яттю або просто кешем (англ .: cache). Кеш-пам'ять зазвичай зроблена з використанням статичної пам'яті (SRAM) і знаходиться в тій же мікросхемі, що і процесор. Швидкість кеша можна порівняти зі швидкістю процесора, так як, по-перше, пам'ять SRAM працює швидше, ніж DRAM, а по-друге, вона знаходиться на кристалі процесора і дозволяє позбутися затримок поширення сигналів по шляху до зовнішніх мікросхем пам'яті.

Якщо процесор запитує дані, які вже знаходяться в кеші, то він отримує їх дуже швидко. Це називається попаданням в кеш (англ .: cache hit). В іншому випадку процесор змушений читати дані з оперативної пам'яті (DRAM). Це називається промахом кеша, кеш-промахом або промахом доступу в кеш (англ .: cache miss). Якщо процесор потрапляє в кеш більшу частину часу, то він рідко простоює в очікуванні доступу до повільної оперативної пам'яті, і середній час доступу малий.

Третій рівень в ієрархії пам'яті - жорсткий диск.

У 2012 році жорсткий диск, зроблений на основі технології магнітного запису (Hard Disk Drive, HDD), мав час доступу близько 10 мілісекунд. Твердотільні диски (Solid State Drive, SSD), які зроблені на основі флеш-пам'яті, стають все більш популярною альтернативою HDD.

Жорсткий диск забезпечує ілюзію наявності більшого обсягу пам'яті, ніж реально є в оперативній пам'яті. Це називається віртуальною пам'яттю. Доступ до даних у віртуальній пам'яті займає тривалий час. Оперативна пам'ять, також звана фізичною пам'яттю, містить тільки частину даних, що знаходяться у віртуальній пам'яті, інші дані знаходяться на жорсткому диску. Отже, оперативна пам'ять може розглядатися як кеш-пам'ять для найбільш часто використовуваних даних з жорсткого диска.

На Рис. 2 наведена типова ієрархія пам’яті.

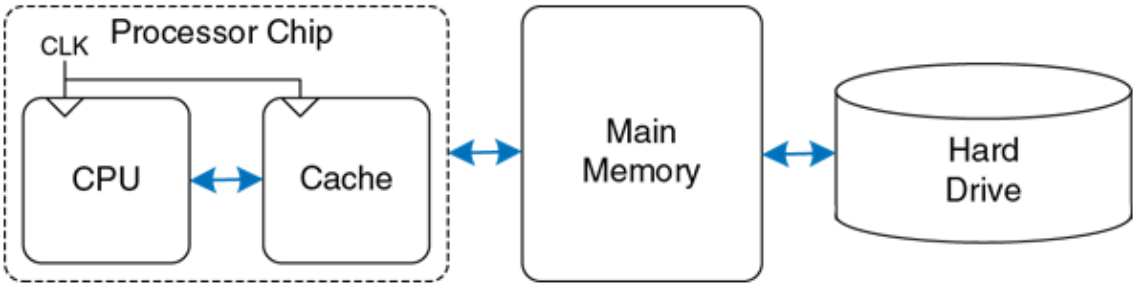
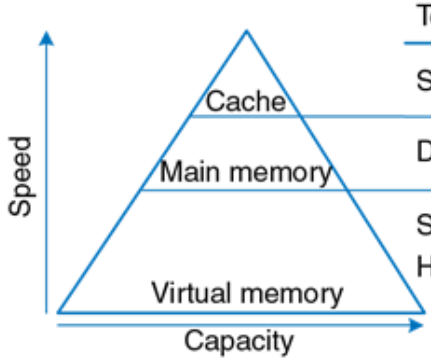


Рис. 2

Процесор спочатку шукає дані в маленькій, але швидкій кеш-пам'яті, зазвичай розташованій на тій же самій мікросхемі. Якщо дані в кеші відсутні, процесор звертається до оперативної пам'яті (Main Memory). Якщо даних немає і там, то процесор читає дані з великого, хоча і повільного, жорсткого диска, використовуючи механізм віртуальної пам'яті. Рис. 3 ілюструє співвідношення ємності і швидкості в багаторівневій ієрархії пам'яті комп'ютерної системи і показує типову вартість, час доступу і пропускну здатність для технологій пам'яті станом на 2012 рік. Як видно, зі зменшенням часу доступу швидкість зростає.



Technology	Price / GB	Access Time (ns)	Bandwidth (GB/s)
SRAM	$10,000	1	25+
DRAM	$10	10–50	10
SSD	$1	100,000	0.5
HDD	$0.1	10,000,000	0.1

Рис. 3

Основні параметри пам'яті

Основними операціями в пам'яті є записування і зчитування певної одиниці інформації, наприклад, байта. Ці операції називаються також зверненням до пам'яті. Пам'ять характеризується інформаційною ємністю, фізичним об'ємом, питомою ємністю і вартістю, шириною вибірки, споживаною потужністю і швидкодією.

Інформаційна ємність E являє собою максимальний об'єм даних, який може одночасно зберігатися в пам'яті. Ємність визначають у бітах, байтах (8 біт = 1 байт), кілобайтах (2^{10}байт = 1 Кбайт), мегабайтах (2^{10} Кбайт = 1 Мбайт) і гігабайтах (2^{10} Мбайт = 1 Гбайт) (при цьому потрібно врахувати, що 2^{10} = 1024). Питома ємність визначається відношенням інформаційної ємності ЗП до його фізичного об'єму.

Питома вартість – це відношення вартості ЗП до його інформаційної ємності.

Споживану потужність задають або для усього ЗП, або на зберігання одного біту інформації. Основними вимогами до пам'яті є максимально велика

інформаційна ємність, висока швидкодія (малий час звернення tзв менший за 10 нс), мінімальна споживана потужність (менша за 1 мкВт на 1 біт інформації, яка зберігається).

У наш час жоден з видів ЗП не задовольняє цих вимог повною мірою. Тому в пам’яті використовуються різні види ЗП, які розрізняються принципами побудови і своїми характеристиками. Швидкодія ЗП вимірюється часом записування і зчитування та тривалістю відповідних їм циклів.

Час записування t_{WR} – це інтервал між моментами появи керуючого сигналу записування і установленням КП в стан, який задають вхідні сигнали.

Час зчитування – це інтервал між моментами появи керуючого сигналу читання t_{RD} і даних на виході пам’яті. Мінімально допустимий інтервал між послідовними читаннями t_{CYR} і записуваннями t_{CYW} створює відповідний цикл.

Тривалість циклів може перевищувати час читання чи записування, оскільки після цих операцій необхідна додаткова затримка для встановлення початкового стану пам’яті. Як тривалість циклу звернення до пам’яті беруть величину $t_{CY} = \max (t_{CYW}, t_{CYR})$.

2. Кеш-пам’ять

Кеш містить часто використовувані дані з пам'яті. Число слів даних, яке він може зберігати, називається ємністю кеша (англ .: capacity). Оскільки ємність кеша менше, ніж ємність оперативної пам'яті, то розробник комп'ютерної системи повинен вирішити, яку підмножину оперативної пам'яті зберігати в кеші.

Коли процесор намагається отримати доступ до даних, він спочатку шукає їх в кеші. Якщо дані там є, тобто відбулося потрапляння в кеш, то процесор отримує їх негайно. Якщо ж їх там немає, тобто стався промах кеша, то процесор зчитує дані з оперативної пам'яті і поміщає їх в кеш для подальшого використання. Для цього кеш повинен замінити якісь старі дані на нові.

Кеш-пам'ять характеризується ємністю C (від англ. capacity), числом наборів S (від англ. set), довжиною рядка, іноді званої розміром блоку b (від англ. block), кількістю рядків або блоків B і ступенем асоціативності N.

Які дані зберігаються в кеш-пам'яті?

Ідеальний кеш повинен передбачати, які дані знадобляться процесору, і вибирати їх з оперативної пам'яті заздалегідь таким чином, щоб кеш мав нульовий відсоток промахів. Але оскільки точно передбачити майбутнє неможливо, то кеш повинен вгадувати, які дані знадобляться, грунтуючись на попередніх зверненнях в пам'ять. Зокрема, кеш використовує часову і просторову локальність, щоб зменшити відсоток промахів в кеш.

Часова локальність означає, що процесор, ймовірно, ще раз звернеться до тих даних, які він недавно використовував. Тому, коли процесор читає або записує дані, яких немає в кеші, то ці дані копіюються з оперативної пам'яті в кеш, так що подальші звернення до них вже не викличуть промаху кеша.

Просторова локальність означає, що коли процесор звертається до будь-яких даних, то, ймовірно, йому знадобляться і розташовані поруч дані. Тому,

коли кеш читає одне слово даних з пам'яті, він заодно читає і кілька сусідніх слів. Ця група слів називається рядком кеша (англ .: cache line), також іноді використовують термін «блок кешу» (англ .: cache block). Число слів в рядку b називається довжиною рядка (line size або block size). Кеш ємністю C містить B = C / b рядків.

Принципи просторової та часової локальності даних були експериментально підтверджені на реальних програмах. Якщо змінна використовується в програмі, то та ж сама змінна буде, швидше за все, використана знову, тим самим створюючи часову локальність. Якщо використовується один елемент масиву, то, швидше за все, і інші елементи цього масиву теж будуть використані, тим самим створюючи просторову локальність.

Як знайти дані в кеш-пам'яті?

Кеш складається з S наборів, кожен з яких містить один або кілька рядків (блоків даних). Взаємозв'язок між адресою даних в оперативній пам'яті і розташуванням цих даних в кеші називається відображенням. Кожна адреса пам'яті завжди відображається в один і той же набір кеша. Кілька біт адреси використовуються, щоб визначити, який саме набір кешу містить потрібні дані. Якщо в наборі більше одного рядка, то дані можуть знаходитися в будь-якому з них.

Кеш-пам'ять класифікується за кількістю рядків у наборі. У кеші *прямого відображення* (англ .: direct mapped cache) кожен набір містить тільки один рядок (один блок), так що кеш містить S = B наборів. Таким чином, кожна з адрес в оперативній пам'яті відображається в один-єдиний рядок кеша. У випадку *набірно-асоціативного* кеша з N секціями (англ .: N-way set associative cache) кожен набір складається з N рядків. Кожна адреса пам'яті як і раніше відображається в один-єдиний набір, але число наборів в цьому випадку дорівнює S = B / N, а дані можуть виявитися в будь-якому з N рядків цього набору. На відміну від кеша прямого відображення і набірно-асоціативного кеша, повністю асоціативний кеш (англ .: fully associative cache) має тільки один набір (S = 1), і дані можуть виявитися в будь-якому з B рядків цього набору. Таким чином, повністю асоціативний кеш - це те ж саме, що і складально-асоціативний кеш з B секціями (кількість секцій збігається з кількістю рядків у всьому кеші).

Кеш-пам'ять прямого відображення

У кеш-пам'яті прямого відображення кожен набір містить тільки один рядок (блок даних), так що у нього S = B наборів і рядків. Щоб зрозуміти спосіб відображення адрес пам'яті в певні рядки такого кеша, уявіть, що оперативна пам'ять поділена на блоки по b слів так само, як кеш поділений на рядки по b слів. Адреса одного з слів, що знаходяться в блоці 0 оперативної пам'яті, відображається в набір 0 кеша. Адреса слова з блоку 1 оперативної пам'яті відображається в набір 1 кешу, і так далі, поки адреса слова з блоку B - 1 оперативної пам'яті не відобразиться в рядок B - 1 кеша. Більше рядків в кеші немає, так що в наступному блоці оперативної пам'яті (блок B) знову з'явиться рядок 0 кешу, і так далі.

Відображення для такого кеша ємністю 8 слів і розміром рядка в одне слово показано на Рис. 3. У кеші 8 наборів, кожен з яких містить по одному рядку, довжина яких дорівнює одному слову.

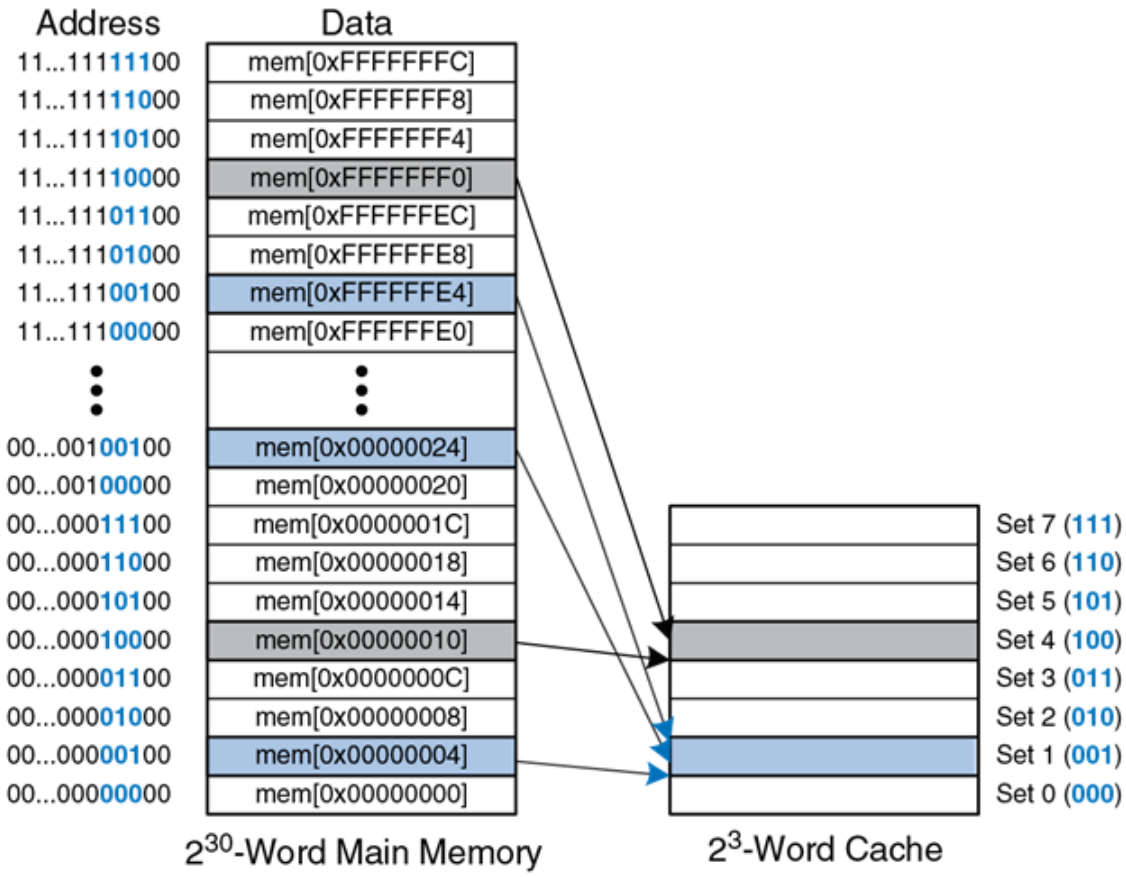


Рис. 3

Так як в один набір кешу відображається безліч адрес, то кеш повинен відслідковувати адреси даних, що знаходяться в кожному з наборів в поточний момент часу. Молодші біти адреси визначають набір, в якому зберігаються дані. Решта біт адреси називаються тегом (англ .: tag) і вказують, яка саме з усіх можливих адрес зараз знаходиться в цьому наборі.

Два молодших біта адреси називаються байтовим зміщенням (byte offset), оскільки вони вказують на номер байта всередині слова. Наступні три біти називаються індексом (cache index) або номером набору (set bits), так як вони вказують на номер набору, в який відображається ця адреса (в загальному випадку номер набору складається з log2S бітів, де S - число наборів). Решта 27 біт тега вказують на адресу слова, яке в поточний момент знаходиться в цьому наборі кеша. На Рис. 4 показано, на які частини ділиться адреса 0xFFFFFFE4.

Рис. 4

Іноді, особливо коли комп'ютер тільки включили, набори кеша ще не містять ніяких даних. Для кожного набору в кеші є біт достовірності (valid bit),

який дорівнює одиниці, якщо в ньому знаходяться коректні дані, і нулю, якщо значення, що знаходиться в ньому є некоректними.

На Рис. 5 зображена блок-схема апаратної реалізації кешу прямого відображення, показаного на Рис. 3. Кеш використовує блок статичної пам'яті SRAM з вісьмома комірками. Кожна комірка, або набір (Set), містить 32 біти даних (Data), 27 бітів тега (Tag) і 1 біт достовірності (V). До кешу звертаються, використовуючи 32-бітові адреси. Два молодших біти адреси - байтове зміщення - ігноруються при зверненні до слів. Наступні 3 біта вказують на комірку, або набір, кеш-пам'яті. Команда завантаження читає цю комірку з кеш-пам'яті і перевіряє тег і біт достовірності. Якщо тег збігається зі старшими 27 бітами адреси і біт достовірності дорівнює 1, то відбувається потрапляння в кеш (Hit) і дані передаються процесору. В іншому випадку відбувається промах кеша і підсистема пам'яті повинна прочитати запитані дані з оперативної пам'яті.

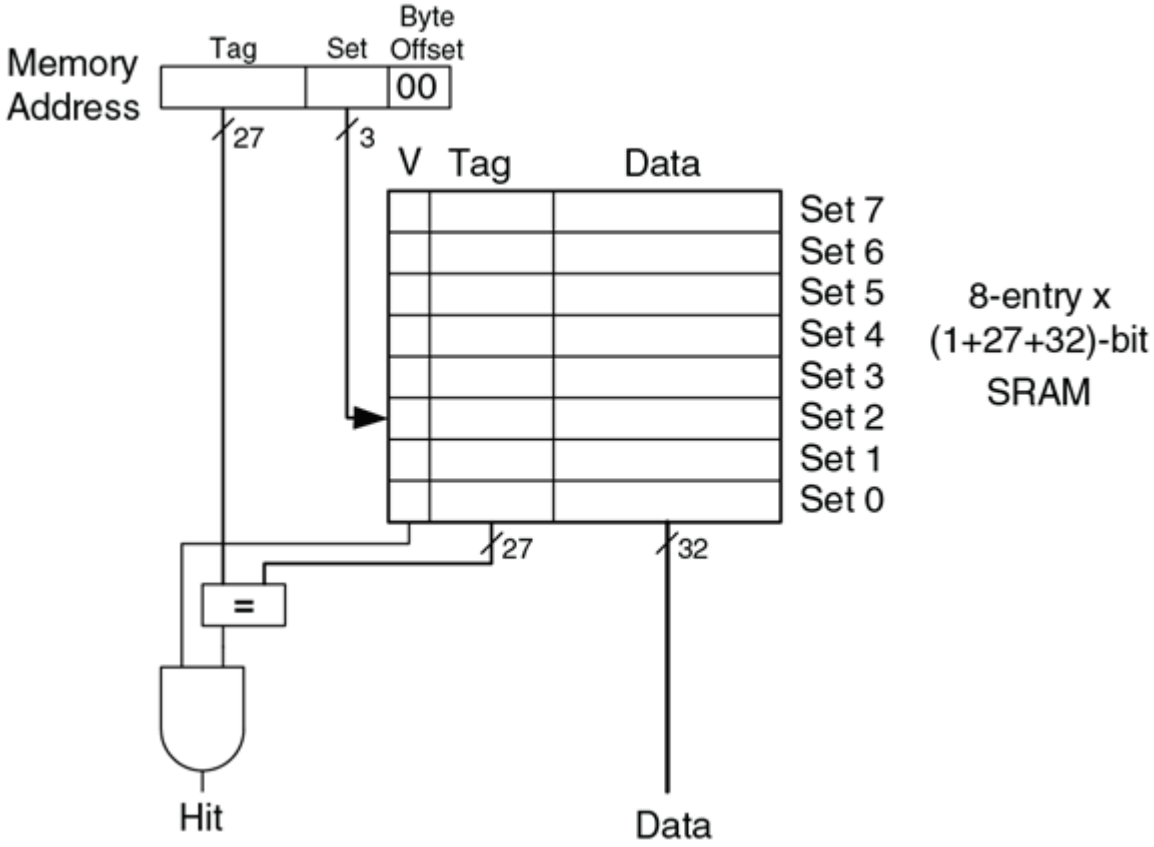


Рис. 5

Багатосекційний набірно-асоціативний кеш

N-секційний набірно-асоціативний кеш (англ .: N-way set associative cache) зменшує число конфліктів шляхом розширення набору до N секцій. Кожна адреса пам'яті як і раніше відображається в строго певний набір, але тепер він може бути відображений в будь-яку з N секцій цього набору. Можна сказати, що кеш прямого відображення - це односекційний набірно-асоціативний кеш. Число N називають ступенем асоціативності кеша.секцій

На Рис. 6 показана блок-схема апаратної реалізації наборно- асоціативного кеша ємністю C = 8 слів, з N = 2 секціями. Тепер в кеші тільки S=4 наборів. Кожен набір тепер містить дві секції (2-way). Кожна секція складається з рядка

(блоку даних), тега і біта достовірності. Кеш читає теги і біти достовірності одночасно з обох секцій обраного набору, після чого порівнює їх з адресою для визначення попадання або промаху. Якщо відбувається попадання в одну із секцій кешу, то мультиплексор вибирає дані з цієї секції і передає їх процесору.

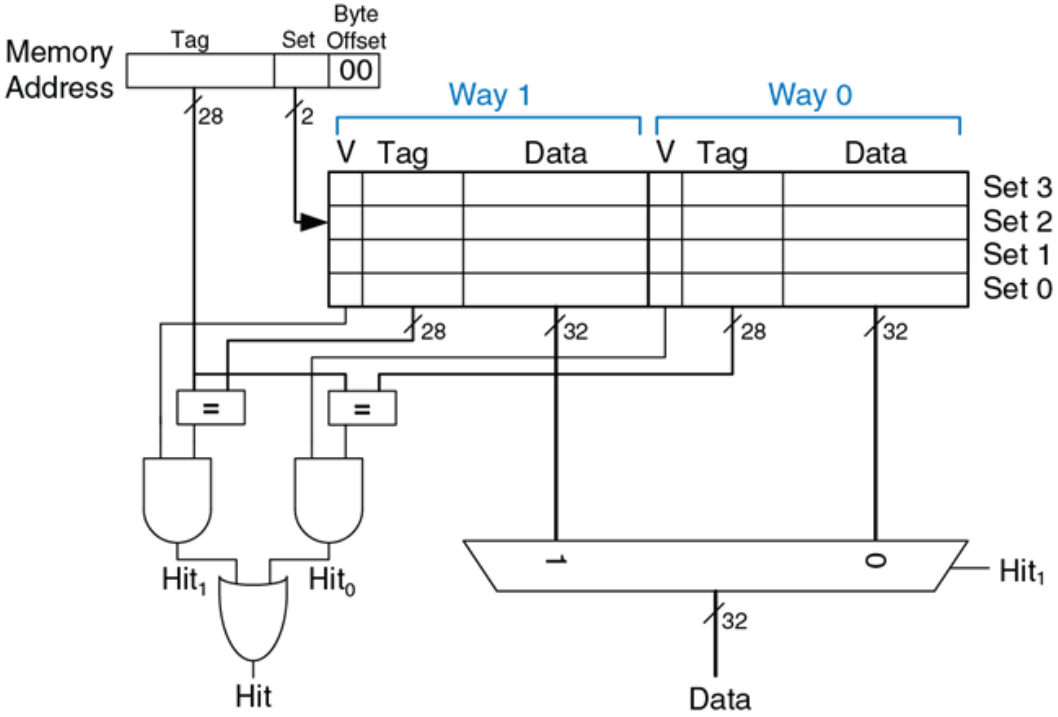


Рис. 6 Двохсекційний наборно-асоціативний кеш

Довжина рядка

Щоб скористатися просторової локальністю, в кеш-пам'яті використовують більші за розміром рядки, що містять кілька послідовних слів.

Переваги рядків з довжиною, що перевищує одне слово, полягає в тому, що коли трапляється промах кеша і потрібно прочитати слово даних з пам'яті, то в цей рядок заодно завантажуються і сусідні слова. Таким чином, наступні звернення з більшою ймовірністю призведуть до потрапляння в кеш через просторову локальність даних.

Які дані замінити в кеш-пам'яті?

У кеш-пам'яті прямого відображення кожна адресу завжди відображається в один і той же рядок одного і того ж набору, тому коли потрібно завантажити нові дані в набір, який вже містить дані, то рядок в наборі просто заміщається на нові дані. У набірно-асоціативній і повністю асоціативній кеш-пам'яті потрібно вирішити, який саме з кількох рядків у наборі витіснити. З огляду на принцип часової локальності, найкращим варіантом було б замінити той рядок, який найдовше не використовувався, тому що малоймовірно, що він буде використаний знову. Саме тому більшість кешів використовують стратегію заміни рідко використовуваних даних (англ .: least recently used, LRU).

У двосекційному набірно-асоціативному кеші біт використання U (від англ. Used) містить номер тієї секції в наборі, яка довше не використовувалася.

Кожен раз, коли відбувається доступ до однієї із секцій набору, біт U встановлюється таким чином, щоб вказувати на іншу секцію.

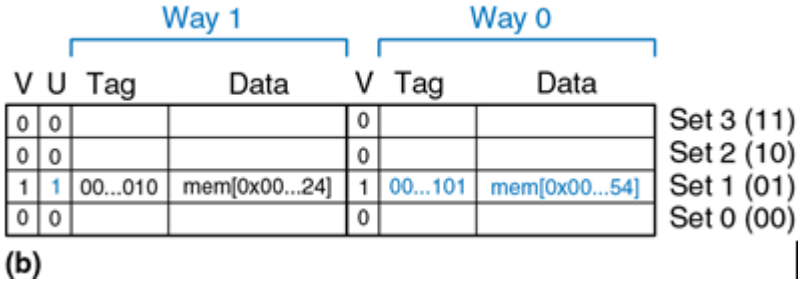


Рис. 7 Двосекційний кеш зі стратегією заміщення LRU

3. Віртуальна пам'ять

У порівнянні з ідеальною пам'яттю, яка повинна бути швидкою, дешевою і великою, жорсткий диск має великий обсяг і недорого коштує, однак неймовірно повільно працює. Жорсткий диск забезпечує набагато більший обсяг, ніж недорога оперативна пам'ять (DRAM). Однак якщо істотна частина звернень до пам'яті здійснюється до жорсткого диска, швидкість роботи сильно знижується.

Мета включення жорсткого диска в ієрархію пам'яті - недорого створити видимість пам'яті великого обсягу, одночасно забезпечуючи для більшості звернень до пам'яті швидкість доступу, рівну швидкості більш швидких типів пам'яті. Наприклад, комп'ютер може забезпечити видимість наявності оперативної пам'яті, використовуючи для цього жорсткий диск.

У цьому випадку велика пам'ять жорсткого диска називається віртуальною пам'яттю, а менша оперативна пам'ять називається фізичною пам'яттю.

Програми можуть звертатися до даних в будь-якому місці віртуальної пам'яті, тому вони повинні використовувати віртуальні адреси, які визначають розташування даних у віртуальній пам'яті. Фізична пам'ять зберігає останні запитані з віртуальної пам'яті блоки даних. Таким чином, фізична пам'ять виступає в ролі кеша для віртуальної пам'яті, тобто більшість звернень відбувається до швидкої фізичної пам'яті (DRAM), і в той же час програма має доступ до більшої за обсягом віртуальної пам'яті.

Віртуальна пам'ять розділена на віртуальні сторінки. Фізична пам'ять аналогічним чином розділена на фізичні сторінки. Розмір віртуальних і фізичних сторінок однаковий. Віртуальна сторінка може розташовуватися в фізичній пам'яті (DRAM) або на жорсткому диску. На Рис. 8. показана віртуальна пам'ять, яка більше фізичної пам'яті. Прямокутники позначають сторінки. Деякі віртуальні сторінки розташовані у фізичній пам'яті, а деякі - на жорсткому диску. Процес перетворення віртуальної адреси у фізичну називається трансляцією адреси.

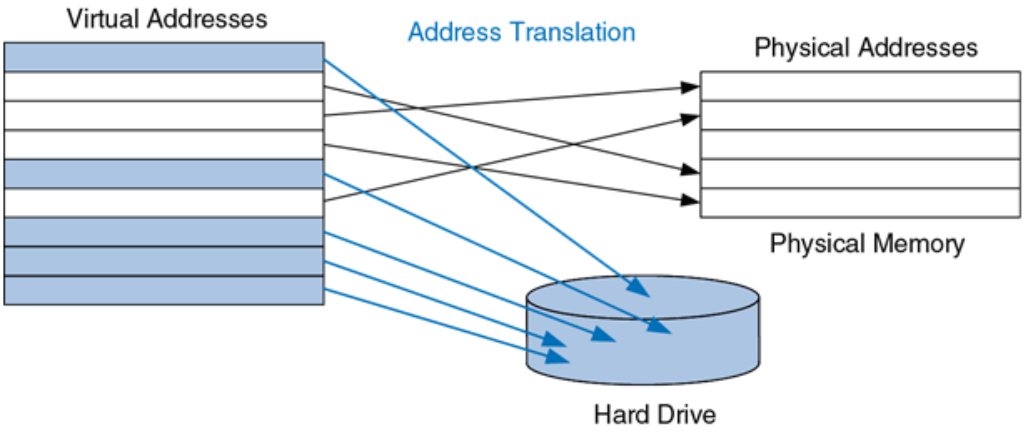


Рис. 8

В підсистемах віртуальної пам'яті використовується трансляція адреси за допомогою таблиці сторінок (англ .: page table). Таблиця сторінок містить запис для кожної віртуальної сторінки, вказує її розташування в фізичній пам'яті або на жорсткому диску. Кожна команда завантаження або збереження вимагає доступу до таблиці сторінок з подальшим доступом до фізичної пам'яті. Звернення до таблиці сторінок дозволяє транслювати віртуальну адресу, що використовується програмою, в фізичну адресу. Потім фізична адреса використовується для фактичного читання або запису даних.

Таблиця сторінок зазвичай настільки велика, що сама знаходиться у фізичній пам'яті. Таким чином, кожна команда завантаження або збереження даних включає два звернення до фізичної пам'яті: звернення до таблиці сторінок і власне звернення до даних. Щоб прискорити трансляцію адреси, використовується буфер асоціативної трансляції (англ .: translation lookaside buffer, TLB), який містить найбільш часто використовувані записи таблиці сторінок.

Таблиця сторінок може зберігатися в будь-якому місці фізичної пам'яті, її розташування визначається операційною системою. Процесор зазвичай використовує виділений регістр, названий регістром таблиці сторінок, для зберігання її базової адреси.

Щоб виконати операцію завантаження або збереження даних, процесор повинен спочатку транслювати віртуальну адресу у фізичну, а потім звернутися до фізичної пам'яті, використовуючи отриману фізичну адресу.

Віртуальна пам'ять мала б величезний негативний вплив на продуктивність, якби було потрібно звернення до таблиці сторінок при виконанні кожної команди завантаження або збереження - це подвоювало б час виконання цих команд. На щастя, звернення до таблиці сторінок мають високу тимчасову локальність. Тимчасова і просторова локальність звернень до даних і великий розмір сторінки означають, що з великою ймовірністю більшість команд завантаження або збереження, які слідують одна за одною, звертаються до однієї і тієї ж сторінки. Тому, якщо процесор запам'ятає останній запис таблиці сторінок, яку він прочитав, то він, швидше за все, зможе повторно використовувати результат трансляції, не виконуючи повторне читання таблиці сторінок.

Системи віртуальної пам'яті забезпечують захист пам'яті, надаючи кожній програмі персональний віртуальний адресний простір ВАП. Кожна програма може використовувати стільки пам'яті в межах віртуального простору, скільки необхідно, але тільки частина віртуального адресного простору знаходиться у фізичній пам'яті в кожен момент часу. Кожна програма може повністю використовувати своє ВАП, не турбуючись про те, де розташовані інші програми. Однак програма має доступ тільки до тих фізичним сторінок, інформація про які знаходиться в її таблиці сторінок. Таким чином, програма не може випадково або навмисно отримати доступ до фізичних сторінок іншої програми, оскільки вони не відображені в її таблиці сторінок. Іноді кілька програм повинні мати доступ до загальних команд або даними. Для цього операційна система додає до кожного запису в таблиці сторінок кілька службових бітів, що дозволяють визначити, які саме програми можуть писати в загальні (колективні, англ .: shared) фізичні сторінки.