## 4.3 КЕШУВАННЯ ДАНИХ

### 4.3.1 Ієрархія запам'ятовуючих пристроїв

Пам'ять обчислювальної машини є ієрархією запам'ятовуючих пристроїв (внутрішні регістри процесора, різні типи надоперативної і оперативної пам'яті, диски), які відрізняються середнім часом доступу і вартістю зберігання даних з розрахунку на один біт (рис 4.4).

Фундаментом цих пристроїв служить зовнішня пам'ять, що, як правило, представляється жорстким диском, який частенько ще називають вінчестер. Вона має великий об'єм (сотні і тисячі гігабайт), але швидкість доступу до даних є невисокою. Час доступу до диска вимірюється мілісекундами.

Принципи сучасної технології виготовлення жорсткого диска були розроблені в 1973 році американською фірмою IBM. Новий пристрій, який міг зберігати до 16 кілобайт інформації, мав 30 циліндрів (доріжок) для запису, кожен з яких був розбитий на 30 секторів. Тому пристрій одержав назву 30/30. Відомі рушниці «вінчестер» мають калібр 30/30, тому, за однією з версій, жорсткі диски теж стали називатися «військовим» словом – «вінчестерами». За іншою, ймовірнішою версією (Велика енциклопедія Кирила і Мефодія, 2005), він одержав таку назву за містом Вінчестер (Winchester, Англія), де проходили первинні розробки жорсткого диска.

На наступному рівні розташовується більше швидкодіюча (час доступу дорівнює приблизно 10-20 наносекундам) і менш об'ємна (від десятків Мегабайт до декількох Гігабайт) оперативна пам'ять, що реалізовується на відносно повільній динамічній пам'яті DRAM. Її часто називають ОЗП (Оперативний Запам'ятовуючий Пристрій, в англомовній літературі RAM – Random Access).

Для зберігання даних, до яких необхідно забезпечити швидкий доступ, використовуються компактні швидкодіючі пристрої на основі статичної пам'яті SRAM, об'єм яких складає від декількох десятків до декількох сотень кілобайт, а час доступу до даних не перевищує 6 нс.

І нарешті, верхівку в цій ієрархії складають внутрішній кеш і внутрішні регістри процесора, які також можуть бути використані для проміжного зберігання даних. Загальний об'єм регістрів складає декілька десятків байт, а час доступу визначається швидкодією процесора і рівний нині приблизно 0,1-0,3 нс. Усі названі

характеристики запам'ятовуючих пристроїв швидко змінюються в міру вдосконалення обчислювальної апаратури. У даному випадку важливі не абсолютні значення часу доступу або об'єму пам'яті, а їх співвідношення для різних типів запам'ятовуючих пристроїв.

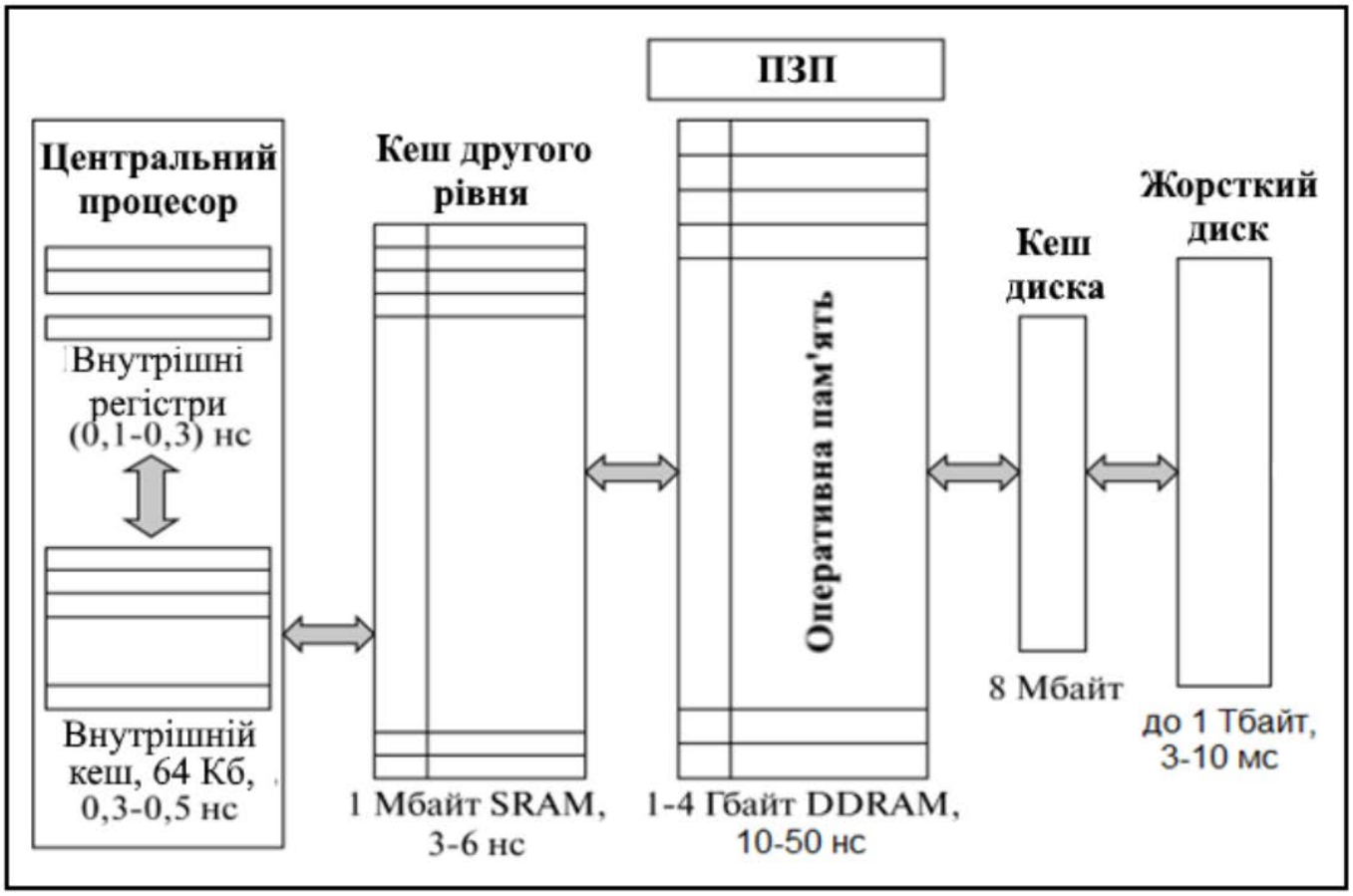


Рисунок 4.4 – Ієрархія пам'яті

Таким чином, можна констатувати закономірність – чим більше об'єм запам'ятовуючого пристрою, тим менш швидкодіючим він є. Більше того, вартість зберігання даних з розрахунку на один біт також збільшується із зростанням швидкодії пристроїв. Проте користувачеві хотілося б мати і недорогу і швидку пам'ять. Кеш-пам'ять представляє деяке компромісне рішення цієї проблеми.

### 4.3.2 Кеш-пам'ять

Кожен з нас використовує кешування у своєму повсякденному житті. У загальних рисах, кеш – це легкодоступне місце для зберігання необхідних речей. Ми

зберігаємо олівці, ручки і скріпки для паперів в ящиках нашого письмового столу, отже ми легко можемо їх узяти, коли вони нам знадобляться.

**Кеш-пам'ять –** це спосіб організації спільного функціонування двох типів запам'ятовуючих пристроїв, які відрізняються часом доступу і вартістю зберігання даних. Цей спосіб дозволяє зменшити середній час доступу до даних за рахунок динамічного копіювання в «швидкий» запам'ятовуючий пристрій (ЗП) інформації з «повільного» ЗП, яка найчастіше використовується.

Кеш-пам'яттю, або кешем, також часто називають не лише спосіб організації роботи двох типів запам'ятовуючих пристроїв, але і один з пристроїв – «швидкий» ЗП. Він коштує дорожче і, як правило, має порівняно невеликий об'єм.

Кешування – це універсальний метод, придатний для прискорення доступу до оперативної пам'яті, до диска і до інших видів запам'ятовуючих пристроїв. Якщо кешування застосовується для зменшення середнього часу доступу до оперативної пам'яті, то в якості кеша використовують швидкодіючу статичну пам'ять. Якщо кешування використовується системою введення-виведення для прискорення доступу до даних, що зберігаються на диску, то в цьому випадку роль кеш-пам'яті виконують буфери в оперативній пам'яті, в яких осідають найактивніше використовувані дані.

### 4.3.3 Принцип дії кеш-пам'яті

Розглянемо окремий випадок використання кеш-пам'яті для зменшення середнього часу доступу до даних, що зберігаються в оперативній пам'яті. Для цього між процесором і оперативною пам'яттю поміщається швидкий ЗП, такий, що називається просто кеш-пам'яттю (рис. 4.5).

Вміст кеш-пам'яті є сукупністю записів про всі завантажені в неї елементи даних. Кожен запис про елемент даних включає:

- значення елементу даних;
- адреса, яку цей елемент даних має в оперативній пам'яті;
- управляючу інформацію: ознака модифікації і ознака звернення до даних за деякий останній період часу.

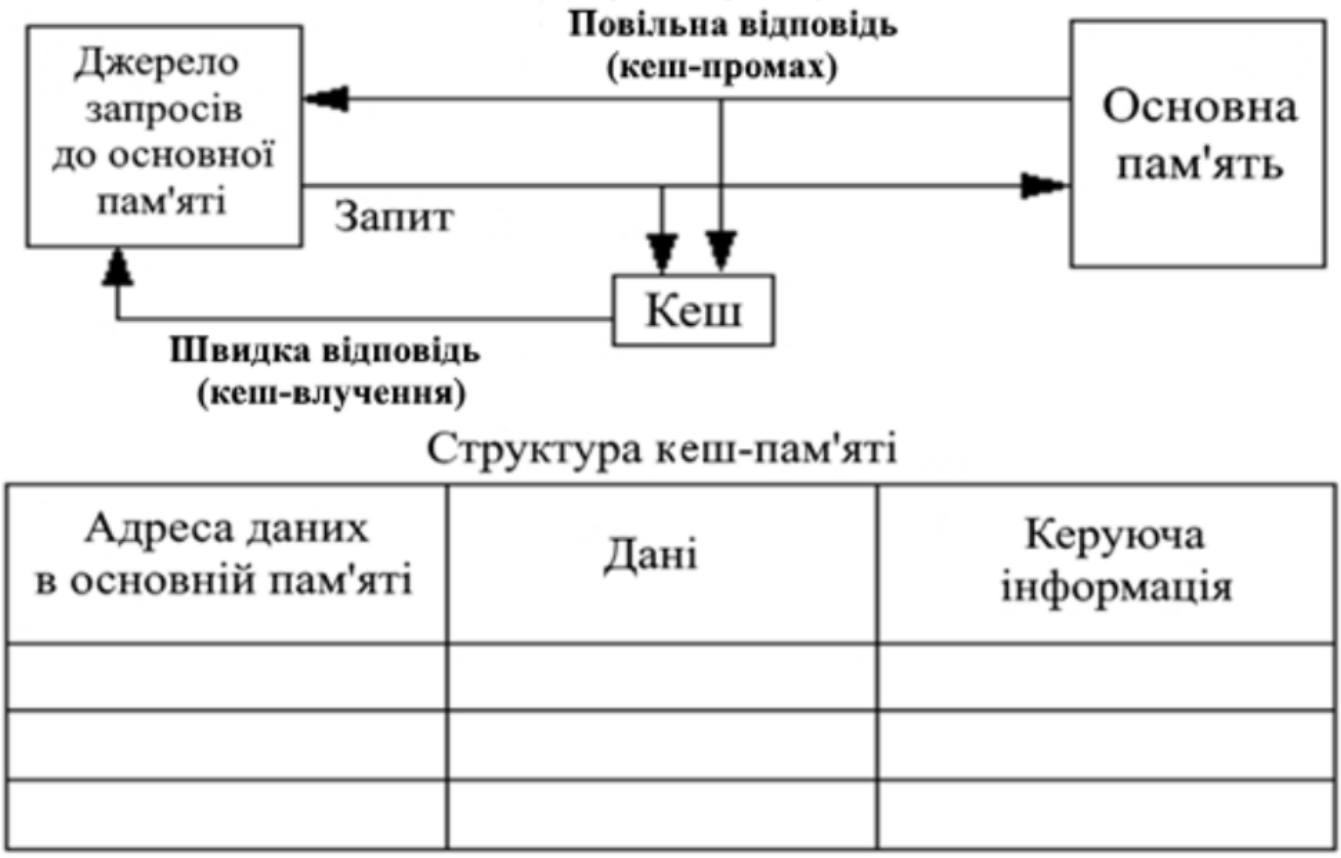


Рисунок 4.5 – Схема функціонування кеш-пам'яті

У системах, оснащених кеш-пам'яттю, кожен запит до оперативної пам'яті виконується відповідно до такого алгоритму.

1. Проглядається вміст кеш-пам'яті з метою визначення, чи не знаходяться потрібні дані в кеш-пам'яті. Кеш-пам'ять не адресується, тому пошук потрібних даних здійснюється по вмісту – значенню поля «адреса в оперативній пам'яті», взятому із запиту.

2. Якщо дані виявляються в кеш-пам'яті, тобто сталося кеш-попадання (cache-hit), то вони прочитуються з неї, і результат передається джерелу запиту (наприклад, процесору).

3. Якщо потрібних даних немає в кеш-пам'яті, тобто стався кеш-промах (cache-miss), то вони разом зі своєю адресою копіюються з оперативної пам'яті в кеш-пам'ять, і результат виконання запиту передається джерелу запиту. При копіюванні даних може виявитися, що в кеш-пам'яті немає вільного місця, тоді вибираються дані, до яких в останній період було менше всього звернень (чи за допомогою іншого алгоритму), для витіснення їх з кеш-пам'яті.

4. Якщо дані, що витісняються, були модифіковані за час знаходження в кеш-пам'яті, то вони переписуються в оперативну пам'ять. Якщо ж ці дані не були модифіковані, то їх місце в кеш-пам'яті оголошується вільним.

На практиці в кеш-пам'ять прочитується не один елемент даних, до якого сталося звернення, а цілий блок даних. Це збільшує ймовірність так званого «попадання в кеш», тобто знаходження потрібних даних в кеш-пам'яті.

Інтуїтивно зрозуміло, що ефективність кешування залежить від ймовірності попадання в кеш. Покажемо це шляхом знаходження залежності середнього часу доступу до основної пам'яті від ймовірності кеш-влучень. Нехай є основний запам'ятовуючий, пристрій з середнім часом доступу до даних tl і кеш-пам'ять, що має час доступу t2, очевидно, що t2 < tl. Нехай t – середній час доступу до даних в системі з кеш-пам'яттю, а p – ймовірність кеш-влучення. За формулою повної ймовірності маємо [19]:

$$t = t1(1 – p) + t2p.$$

Середній час доступу до даних в системі з кеш-пам'яттю лінійно залежить від ймовірності попадання в кеш і змінюється від середнього часу доступу в основний запам'ятовуючий пристрій t1 при p=0 до середнього часу доступу безпосередньо в кеш-пам'ять t2 при p=1. Звідси видно, що використання кеш- пам'яті має сенс тільки при високій ймовірності кеш-влучення.

Ймовірність виявлення даних в кеші залежить від різних чинників, таких, наприклад, як обсяг кешу, обсяг кешованої пам'яті, алгоритму заміщення даних в кеші, особливості виконуваної програми, час її роботи, рівень мультипрограмування і інших особливостей обчислювального процесу. Проте в більшості реалізацій кеш-пам'яті відсоток кеш-влучень виявляється дуже високим – понад 90%. Таке високе значення ймовірності знаходження даних в кеш-пам'яті пояснюється наявністю в них об'єктивних властивостей: просторової і часової локальності.

Просторова локальність. Якщо сталося звернення за деякою адресою, то з високою ймовірністю найближчим часом станеться звернення до сусідніх адрес.

Часова локальність. Якщо сталося звернення за деякою адресою, то наступне звернення за цією ж адресою з великою ймовірністю станеться найближчим часом.

Саме грунтуючись на властивості часової локальності, дані, тільки що зчитані з основної пам'яті, розміщують в пристрої швидкого доступу, припускаючи, що скоро вони знову знадобляться. Зпочатку роботи системи, коли кеш-пам'ять ще порожня, майже кожен запит до основної пам'яті виконується «за повною програмою»: перегляд кеша, констатація промаху, читання даних з основної пам'яті, передача результату джерелу запиту і копіювання даних в кеш. Потім, у міру заповнення кеша, у повній відповідності з властивістю часової локальності зростає ймовірність звернення до даних, які вже були використані на попередньому етапі роботи системи, тобто до даних, які містяться в кеші і можуть бути зчитані значно швидше, ніж з основної пам'яті.

Властивість просторової локальності також використовується для збільшення ймовірності кеш-попадання. Як правило, в кеш-пам'ять прочитується не один інформаційний елемент, до якого сталося звернення, а цілий блок даних, розташованих в основній пам'яті в безпосередній близькості з цим елементом. Оскільки при виконанні програми дуже висока ймовірність, що команди вибираються з пам'яті послідовно одна за одною з сусідніх комірок, то має сенс завантажувати в кеш-пам'ять цілий фрагмент програми. Аналогічно якщо програма веде обробку деякого масиву даних, то її роботу можна прискорити, завантаживши в кеш частину або навіть увесь масив даних.

### 4.3.4 Проблема узгодження даних

У процесі роботи вміст кеш-пам'яті постійно оновлюється, тобто час від часу дані з неї повинні витіснятися. Витіснення означає або просте оголошення вільної відповідної області кеш-пам'яті (скидання біта дійсності), якщо дані, що витісняються, за час знаходження в кеші не були змінені, або копіювання даних з кешу в основну пам'ять, якщо вони були модифіковані.

Алгоритм заміни даних в кеш-пам'яті істотно впливає на її ефективність. В ідеалі такий алгоритм повинен, по-перше, бути максимально швидким, щоб не уповільнювати роботу кеш-пам'яті, а, по-друге, забезпечувати максимально можливу ймовірність кеш-влучень.

Наявність в комп'ютері двох копій даних – в основній пам'яті і в кеші – породжує проблему узгодження даних. Якщо відбувається запис в основну пам'ять за

деякою адресою, а вміст цієї пам'яті знаходиться в кеші, то в результаті відповідний запис в кеші стає недостовірним. Розглянемо два підходи до вирішення цієї проблеми:

1. Наскрізний запис (write through). При кожному запиті до основної пам'яті, у тому числі і при запису, проглядається кеш. Якщо дані за запрошеною адресою відсутні, то запис виконується тільки в основну пам'ять. Якщо ж дані, до яких виконується звернення, знаходяться в кеші, то запис виконується одночасно в кеш і основну пам'ять.

2. Зворотний запис (write back). При виникненні запиту до пам'яті виконується перегляд кеша, і якщо запрошених даних там немає, то запис виконується тільки в основну пам'ять. В іншому ж випадку запис проводиться тільки в кеш-пам'ять. При цьому в описувачі даних робиться спеціальна відмітка (ознака модифікації), яка вказує на те, що при витісненні цих даних з кеша необхідно переписати їх в основну пам'ять, щоб актуалізувати застарілий вміст основної пам'яті.

У деяких алгоритмах заміщення передбачається першочергове вивантаження модифікованих даних. Модифіковані дані можуть вивантажуватися не лише при звільненні місця в кеш-пам'яті для нових даних, але і в «фоновому режимі», коли система не дуже завантажена.

### 4.3.5 Способи відображення основної пам'яті на кеш

Алгоритм пошуку і алгоритм заміщення даних в кеші безпосередньо залежать від того, яким чином основна пам'ять відображається на кеш-пам'ять. Принцип прозорості вимагає, щоб правило відображення основної пам'яті на кеш-пам'ять не залежало від роботи програм і користувачів. При кешуванні даних широко використовуються дві основні схеми відображення: випадкове відображення і детерміноване відображення.

При випадковому відображенні елемент оперативної пам'яті в загальному випадку може бути розміщений в довільному місці кеш-пам'яті. Для того щоб надалі можна було знайти потрібні дані в кеші, вони поміщаються туди разом зі своєю адресою, тобто з тією адресою, яку дані мають в оперативній пам'яті. При кожному запиті до оперативної пам'яті виконується пошук в кеші, причому критерієм пошуку виступає адреса оперативної пам'яті із запиту. Очевидна схема простого перебору для пошуку потрібних даних у разі кеша виявляється непридатною із-за неприпустимо

великих часових витрат. Для кешів з випадковим відображенням використовується так званий асоціативний пошук, при якому порівняння виконується не послідовно з кожним записом кеша, а паралельно з усіма його записами (рис. 4.6).


Рисунок 4.6 – Асоціативний пошук в кеші з випадковим відображенням

Ознака, за якою виконується порівняння, називається тегом (tag). В даному випадку тегом є адреса даних в оперативній пам'яті. Електронна реалізація такої схеми призводить до здорожчання пам'яті, причому вартість істотно зростає зі збільшенням об'єму пристрою кеша. Тому асоціативна кеш- пам'ять використовується в тих випадках, коли для забезпечення високого відсотка попадання достатньо невеликого об'єму пам'яті.

У кешах, побудованих на основі випадкового відображення, витіснення старих даних відбувається тільки в тому випадку, коли вся кеш-пам'ять заповнена і немає вільного місця. Вибір даних на вивантаження здійснюється серед усіх записів кеша. Цей вибір грунтується на тих же прийомах, що і в алгоритмах заміщення сторінок, наприклад вивантаження даних, до яких найдовше не було звернень, або даних, до яких було менше всього звернень.

Детермінований спосіб відображення припускає, що будь-який елемент основної пам'яті завжди відображається в одне і те ж місце кеш-пам'яті. У цьому випадку кеш-пам'ять розділена на рядки, кожний з яких призначений для зберігання одного запису про один елемент даних і має свій номер (насправді запис в кеші містить декілька елементів даних).

Між номерами рядків кеш-пам'яті і адресами оперативної пам'яті встановлюється відповідність «один до багатьох»: одному номеру рядка відповідає декілька адрес оперативної пам'яті.

В якості відображаючої функції може використовуватися просте виділення декількох розрядів з адреси оперативної пам'яті, які інтерпретуються як номер рядка кеш-пам'яті (таке відображення називається прямим). Наприклад, нехай в кеш-пам'яті може зберігатися 1024 записи. Тоді будь-яка адреса оперативної пам'яті може бути відображена на адресу кеш-пам'яті простим відділенням 10 двійкових розрядів (рис. 4.7).

При пошуку даних в кеші використовується швидкий прямий доступ до запису за номером рядка, отриманого шляхом обробки адреси оперативної пам'яті із запиту. Проте оскільки в знайденому рядку можуть знаходитися дані з будь-якого елементу оперативної пам'яті, молодші розряди адреси якої співпадають з номером рядка, необхідно виконати додаткову перевірку. Для цих цілей кожен рядок кеш-пам'яті доповнюється тегом, що містить старшу частину адреси даних в оперативній пам'яті. При співпаданні тега з відповідною частиною адреси із запиту, констатується кеш-попадання.

Якщо ж стався кеш-промах, то дані прочитуються з оперативної пам'яті і копіюються в кеш. Якщо рядок кеш-пам'яті, в яку має бути скопійований елемент даних з оперативної пам'яті, містить інші дані, то останні витісняються з кеша. Відмітимо, що процес заміщення даних в кеш-пам'яті на основі прямого відображення істотно відрізняється від процесу заміщення даних в кеш-пам'яті з випадковим відображенням. По-перше, витіснення даних відбувається не лише у разі відсутності вільного місця в кеші, по-друге, ніякого вибору даних на заміщення не існує.


Рисунок 4.7 – Пряме відображення

У багатьох сучасних процесорах кеш-пам'ять будується на основі поєднання цих двох підходів, що дозволяє знайти компроміс між порівняно низькою вартістю кеша з прямим відображенням і інтелектуальністю алгоритмів заміщення в кеші з випадковим відображенням (рис. 4.8).

При змішаному підході довільна адреса оперативної пам'яті відображається не на одну адресу кеш-пам'яті (як це характерно для прямого відображення), і не на будь-яку адресу кеш-пам'яті (як це робиться при випадковому відображенні), а на деяку групу адрес. Усі групи пронумеровані. Пошук в кеші здійснюється спочатку за номером групи, отриманим з адреси оперативної пам'яті із запиту, а потім в межах групи шляхом асоціативного перегляду усіх записів в групі на предмет співпадання старших частин адрес оперативної пам'яті.

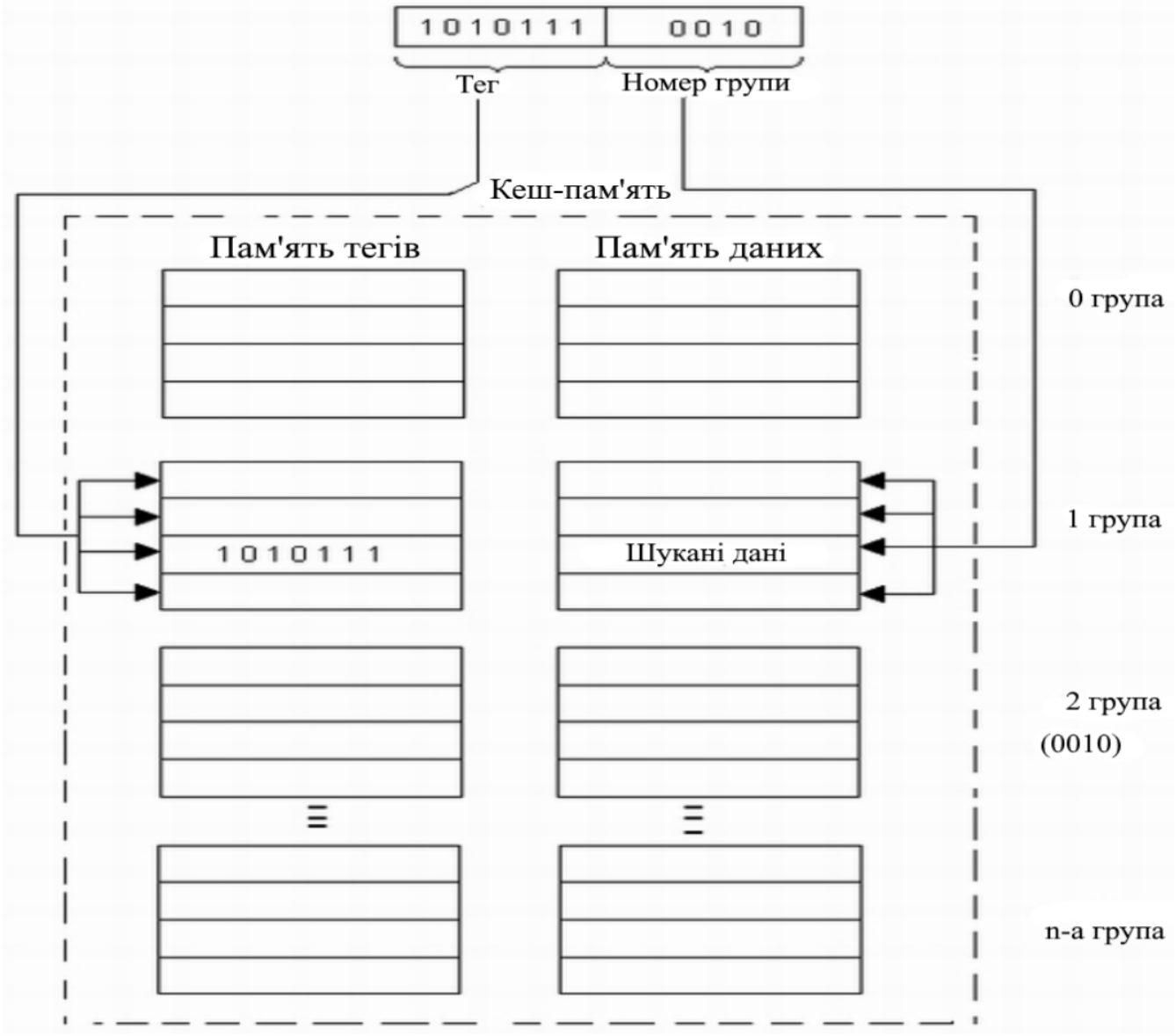


Рисунок 4.8 – Комбінування прямого і випадкового відображення

При промаху дані копіюються за будь-якою вільною адресою з однозначно заданої групи. Якщо вільних адрес в групі немає, то виконується витіснення даних. Оскільки кандидатів на вивантаження декілька – усі записи з цієї групи – алгоритм заміщення може врахувати інтенсивність звернень до даних і тим самим підвищити ймовірність попадань в майбутньому. Таким чином, у цьому способі комбінується пряме відображення на групу і випадкове відображення в межах групи.