10.3 ВІРТУАЛЬНА ПАМ'ЯТЬ. СТОРІНКОВИЙ РОЗПОДІЛ ПАМ'ЯТІ

У наступних розділах мова піде про найпоширенішу нині схему управління пам'яттю, відому як **віртуальна пам'ять**, у рамках якої здійснюється складний зв'язок між апаратним і програмним забезпеченням. Розбиття адресного простору процесу на частини і динамічна трансляція адреси дозволили виконувати процес навіть за відсутності деяких його компонентів в оперативній пам'яті. Наслідком такої стратегії є можливість виконання великих програм, розмір яких може перевищувати розмір оперативної пам'яті.

Вже досить давно користувачі зіштовхнулися з проблемою розміщення в пам'яті програм, розмір яких перевищував вільну пам'ять, що є в наявності. Рішенням було розбиття програми на частини, що називаються оверлеями. 0-ий оверлей починав виконуватися першим. Коли він закінчував своє виконання, він викликав інший оверлей. Усі оверлеї зберігалися на диску і переміщалися між пам'яттю і диском засобами операційної системи. Проте розбиття програми на частини і планування їх завантаження в оперативну пам'ять повинен був здійснювати програміст.

Розвиток методів організації обчислювального процесу в цьому напрямі привів до появи методу, відомого під назвою віртуальна пам'ять. **Віртуальним** називається ресурс, який користувачеві або призначеній для користувача програмі представляється таким, що має властивості, якими він насправді не володіє. Так, наприклад, користувачеві може бути надана віртуальна оперативна пам'ять, розмір якої перевершує усю наявну в системі реальну оперативну пам'ять. Користувач пише програми так, як ніби в його розпорядженні є однорідна оперативна пам'ять великого об'єму, але насправді усі дані, використовувані програмою, зберігаються на одному або декількох різнорідних пристроях, зазвичай на дисках, і при необхідності частинами відображаються в реальну пам'ять.

Уперше віртуальна пам'ять була використана в операційній системі машини Atlas (Англія, 1962 рік) [12]. Незабаром після цього віртуальна пам'ять стала широко використовуватися в комерційних системах. Atlas Supervisor була здатна виконувати до 16 задач одночасно, встановивши новий стандарт продуктивності.

Спираючись на емпіричні (засновані на спостереженнях) результати, можна відмітити, що основний об'єм пам'яті, яку займає процес, велику частину часу залишається вільним, то існує таке правило «дев'яносто до десяти», або правило локалізації (локальності), яке стверджує, що 90% звернень до пам'яті в процесі доводиться на 10% його адресного простору.

Локальність – емпірична закономірність, що зв'язує близькі за часом або в просторі події. Якщо у вашому місті сонячно, то, ймовірно, але не напевно, поблизу від нього теж сонячно. Якщо зараз в місті гарна погода, то, ймовірно, але не напевно, вона була хорошою деякий час назад і залишиться хорошою ще деякий час у майбутньому.

У застосуванні до послідовностей звернень до пам'яті **просторова локальність** означає, що процес, який звертався раніше за деякою адресою, ймовірно, звертатиметься і до близьких до нього адрес. **Часова локальність** означає, що процес, який звертався нещодавно за якоюсь адресою, ймовірно, звернеться до неї знову.

Застосовуючи модель локальності, в основній пам'яті можна зберігати постійно тільки ті розділи адресного простору, які дійсно використовуються в конкретний момент. При цьому невикористані розділи адресного простору можна ставити у відповідність менш швидкій пам'яті, наприклад, простору на жорсткому диску, а в цей час інші процеси можуть використовувати основну пам'ять, займану раніше цими розділами. Якщо ж ці розділи знадобляться знову, то вони можуть бути завантажені з диска в основну пам'ять (може бути в інший адресний простір).

Таким чином, віртуальна пам'ять – це сукупність програмно-апаратних засобів, що дозволяють користувачам писати програми, розмір яких перевищує наявну оперативну пам'ять. Для цього віртуальна пам'ять розв'язує такі задачі:

1. Розміщує дані в пристроях різного типу, наприклад, частина програми в оперативній пам'яті, а частина на диску.

2. Переміщає в міру необхідності дані між пристроями різного типу, наприклад, підвантажує потрібну частину програми з диска в оперативну пам'ять.

3. Перетворює віртуальні адреси у фізичні.

Усі ці дії виконуються автоматично, без участі програміста, тобто механізм віртуальної пам'яті є прозорим стосовно користувача. Найбільш поширеними реалізаціями віртуальної пам'яті є сторінковий, сегментний і сторінково - сегментний розподіл пам'яті, а також свопінг.

10.3.1 Сторінковий розподіл пам'яті

Як розділи з різними фіксованими розмірами, так і розділи змінного розміру недостатньо ефективно використовують оперативну пам'ять. Результатом роботи перших стає внутрішня фрагментація, результатом останніх – зовнішня.

При сторінковій організації віртуальний адресний простір кожного процесу ділиться на частини однакового, невеликого фіксованого для цієї системи розміру, що називаються віртуальними сторінками. У загальному випадку розмір віртуального адресного простору не є кратним розміру сторінки, тому остання сторінка кожного процесу доповнюється фіктивною областю.

Уся оперативна пам'ять машини також ділиться на частини такого ж розміру, що називаються **фізичними сторінками** (або блоками). Вільні блоки оперативної пам'яті ще відомі як **кадри** (frames) або фрейми. Кожен кадр може містити одну сторінку даних.

Розмір сторінки звичайно вибирається рівним степеню двійки: 1024, 2048, 4096 тощо. Це дозволяє спростити механізм перетворення адрес.

При створенні процесу частина його віртуальних сторінок (початкові сторінки кодового сегменту і сегменту даних) поміщається в оперативну пам'ять. Копія усього віртуального адресного простору процесу знаходиться на диску. Частина процесу, яка розташована в деякий момент часу в основній пам'яті, називається резидентною множиною процесу. Суміжні віртуальні сторінки не обов'язково розташовуються в суміжних фізичних сторінках.

Операційна система створює для кожного процесу інформаційну структуру – таблицю сторінок. Кількість дескрипторів у таблиці і їх розмір підбираються такими, щоб об'єм таблиці виявився рівним об'єму сторінки. Адреса таблиці сторінок включається в контекст процесу. При активізації чергового процесу ОС завантажує адресу його таблиці сторінок в спеціальний регістр. У таблиці сторінок

встановлюється відповідність між номерами віртуальних і фізичних сторінок (Nф), які завантажені в оперативну пам'ять, або робиться відмітка, що віртуальна сторінка вивантажена на диск (ВП, рис. 10.8).

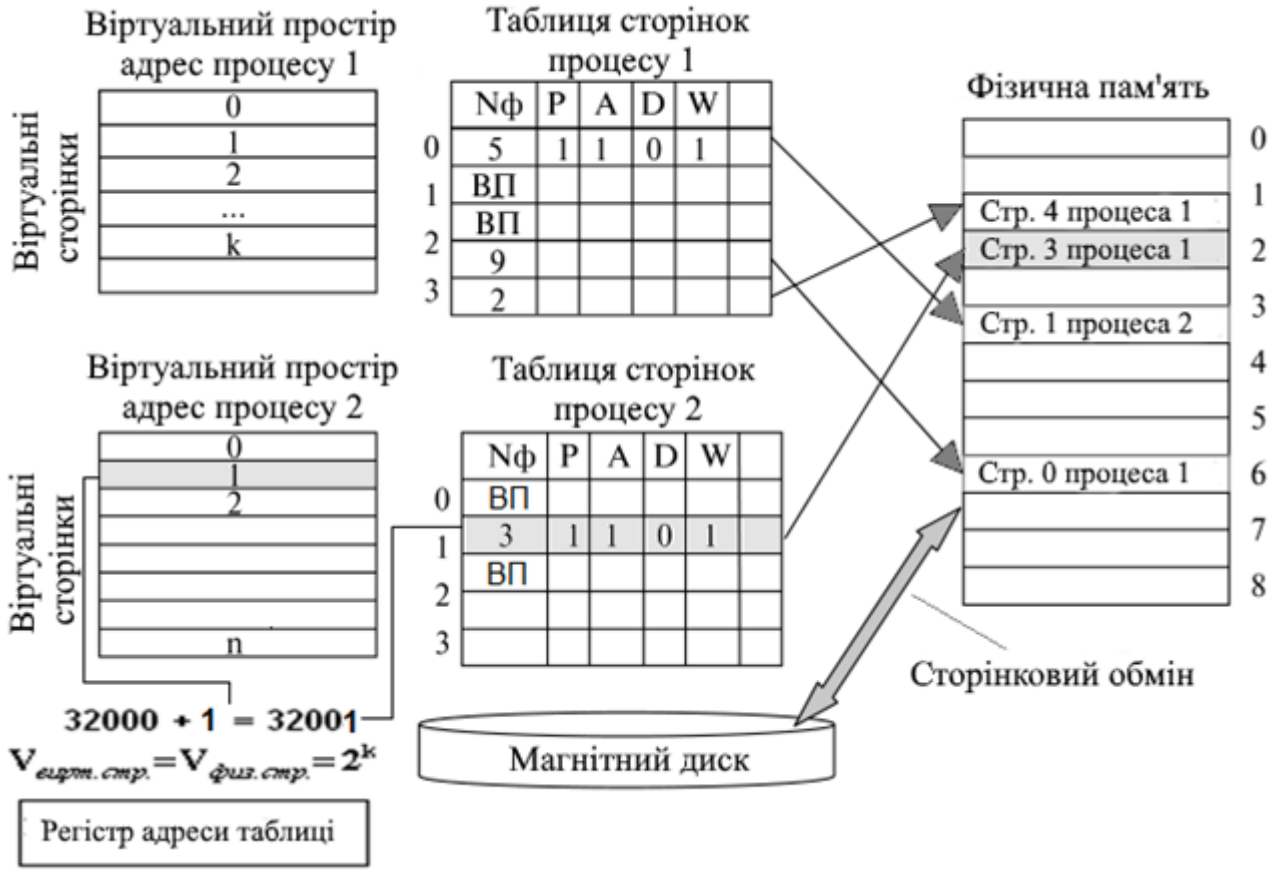

Рисунок 10.8 – Таблиці сторінок віртуальної пам'яті

Крім того, в кожному записі таблиці (дескриптор сторінки) міститься управляюча інформація:

– ознака присутності P, що встановлюється в одиницю, якщо ця сторінка знаходиться в оперативній пам'яті;

– ознака зміни (модифікації) сторінки D, яка встановлюється в одиницю всякий раз, коли робиться запис за адресою, що належить до цієї сторінки;

– ознака звернення A до сторінки, що називається також бітом доступу, який встановлюється в одиницю при кожному зверненні за адресою, що належить до цієї сторінки;

– ознака захисту сторінки W (читання, читання/запис);

– інші управляючі службові біти, наприклад, для цілей захисту або спільного використання пам'яті на рівні сторінок.

Ознаки присутності, модифікації і звернення в більшості моделей сучасних процесорів встановлюються апаратно, схемами процесора при виконанні операції з пам'яттю. Інформація з таблиць сторінок використовується для вирішення питання про необхідність переміщення тієї або іншої сторінки між пам'яттю і диском, а також для перетворення віртуальної адреси у фізичну. Самі таблиці сторінок, також як і описувані ними сторінки, розміщуються в оперативній пам'яті. Адреса таблиці сторінок включається в контекст відповідного процесу. При активізації чергового процесу операційна система завантажує адресу його таблиці сторінок в спеціальний регістр процесора.

При кожному зверненні до пам'яті відбувається читання з таблиці сторінок інформації про віртуальну сторінку, до якої сталося звернення. Якщо ця віртуальна сторінка знаходиться в оперативній пам'яті, то виконується перетворення віртуальної адреси у фізичну. Якщо ж потрібна віртуальна сторінка в даний момент вивантажена на диск, то відбувається так зване сторінкове переривання. Процес, що виконується, переводиться в стан очікування, і активізується інший процес з черги готових.

Паралельно програма обробки сторінкового переривання знаходить на диску необхідну віртуальну сторінку і намагається завантажити її в оперативну пам'ять. Якщо в пам'яті є вільна фізична сторінка, то завантаження виконується негайно, якщо ж вільних сторінок немає, то вирішується питання, яку сторінку слід вивантажити з оперативної пам'яті.

У цій ситуації може бути використано багато різних критеріїв вибору, але найпопулярніші з них такі:

– сторінка, яка найдовше не використалася;

– перша, яка попалась, сторінка;

– сторінка, до якої останнім часом було менше всього звернень.

У деяких системах використовується поняття робочої множини сторінок. Робоча множина визначається для кожного процесу і є переліком найчастіше використовуваних сторінок, які повинні постійно знаходитися в оперативній пам'яті і тому не підлягають вивантаженню.

Після того, як вибрана сторінка, яка повинна покинути оперативну пам'ять, аналізується її ознака модифікації (з таблиці сторінок). Якщо виштовхана сторінка з

моменту завантаження була модифікована, то її нова версія має бути переписана на диск. Якщо ні, то вона може бути просто знищена, тобто відповідна фізична сторінка оголошується вільною. Розглянемо механізм перетворення віртуальної адреси у фізичну при сторінковій організації пам'яті.

Віртуальна адреса при сторінковому розподілі представлена у вигляді пари (P, SV), де P – порядковий номер віртуальної сторінки процесу (нумерація сторінок починається з 0), а SV – зміщення в межах віртуальної сторінки (I – індекс). Фізична адреса також може бути представлена у вигляді пари (N, SF), де N – номер фізичної сторінки, а SF – зміщення в межах фізичної сторінки. Задача підсистеми віртуальної пам'яті полягає у відображенні (P, SV) в (N, SF).

Перш ніж приступити до розгляду схеми перетворення віртуальної адреси у фізичну, зупинимося на двох базисних властивостях сторінкової організації.

Перша з них полягає в тому, що об'єм сторінки вибирається рівним степені двійки – 2^k. З цього виходить, що зміщення S може бути отримане простим відділенням k молодших розрядів в двійковому записі адреси, а старші розряди адреси, що залишилися, є двійковим записом номера сторінки. При цьому неважливо, є сторінка віртуальною або фізичною. Наприклад, якщо розмір сторінки 1 Кб (2^{10}), то з двійкового запису адреси 5071_8 = 101 000 111 001_2 можна визначити, що він належить сторінці в двійковому вираженні 10_2, і зміщений відносно її початку на 1 000 111 0012 байт.

Тобто, номер сторінки і її початкова адреса легко можуть бути отримані один з іншого доповненням або відкиданням k нулів, що відповідають зміщенню.

Саме з цієї причини часто говорять, що таблиця сторінок містить початкову фізичну адресу сторінки в пам'яті (а не номер фізичної сторінки), хоча насправді в таблиці вказані тільки старші розряди адреси. Початкова адреса сторінки називається базовою адресою. Друга властивість полягає в тому, що в межах сторінки безперервна послідовність віртуальних адрес однозначно відображається в безперервну послідовність фізичних адрес, тобто зміщення у віртуальній і фізичній адресах SV і SF рівні між собою (SV = SF).

Звідси слідує проста схема перетворення віртуальної адреси у фізичну (рис. 10.9). Молодші розряди фізичної адреси, що відповідають зміщенню,

створюються перенесенням такої ж кількості молодших розрядів з віртуальної адреси. Старші розряди фізичної адреси, що відповідають номеру фізичної сторінки, визначаються з таблиці сторінок, в якій вказується відповідність віртуальних і фізичних сторінок.

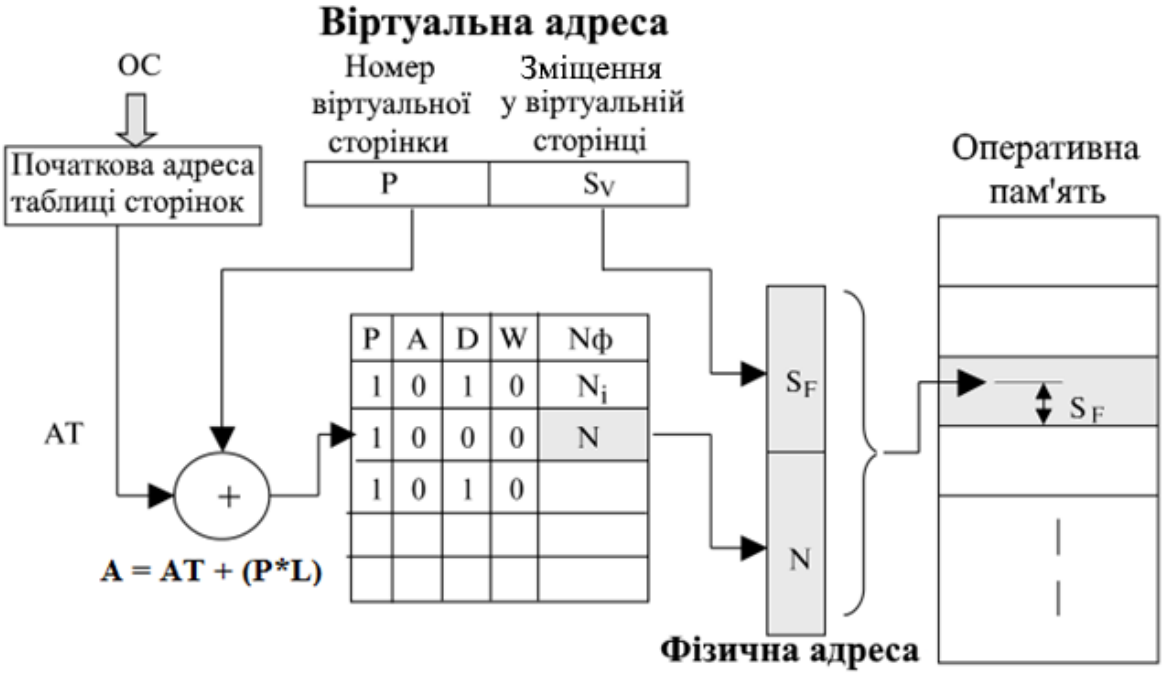

Рисунок 10.9 – Перетворення віртуальної адреси у фізичну при сторінковій організації пам'яті

При кожному зверненні до оперативної пам'яті апаратними засобами виконуються такі дії:

1. Із спеціального регістра процесора витягається адреса AT таблиці сторінок активного процесу. На підставі початкової адреси таблиці сторінок, номери віртуальної сторінки P (старші розряди віртуальної адреси) і довжини окремого запису в таблиці сторінок L (системна константа) визначається адреса потрібного дескриптора в таблиці сторінок: A=AT(P*L).
2. З цього дескриптора витягається номер відповідної фізичної сторінки – N.
3. До номера фізичної сторінки приєднується зміщення SV (молодші розряди віртуальної адреси).

Саме для зменшення часу перетворення адрес в усіх процесорах передбачений апаратний механізм отримання фізичної адреси за віртуальною адресою. З тією ж метою розмір сторінки вибирається рівним степені двійки, завдяки чому двійковий

запис адреси легко розділяється на номер сторінки і зміщення, і в результаті в процедурі перетворення адрес триваліша операція складання замінюється операцією приєднання (конкатенації).

Розглянемо приклад, що пояснює основні характеристики організації сторінкової віртуальної пам'яті. Нехай комп'ютер має оперативну пам'ять об'ємом Еоп = 256 Мб, розмір сторінки вибраний рівним Естр = 4 Кб. У цьому випадку кількість фізичних сторінок рівна

$$Nf = Еоп / Естр = 256*20^{20} / 4*2^{10} = 64000 \text{ сторінок.}$$

Для відображення фізичної адреси довільного байта оперативної пам'яті знадобиться $K = \log_2 256*20^{20} = 28$ двійкових розрядів.

Число розрядів для відображення зміщення в сторінці

$$M = \log_2 4 \text{ Кб} = \log_2 4096 = 12.$$

Якщо процесор має 32-розрядну структуру, то на номер віртуальної сторінки відводиться 32-12=20 двійкових розрядів. Таким чином, число віртуальних сторінок дорівнює $Nв = 2^{20}$ (приблизно 1 млн віртуальних сторінок). На адресу фізичної сторінки в нашому прикладі слід виділити 32-12 = 20 двійкових розрядів. На додаткові розряди, що характеризують властивості сторінки, знадобиться 1 байт. З іншого боку, немає необхідності в записі (дескрипторі) віртуальної сторінки мати поле з номером віртуальної сторінки (20 розрядів), оскільки адресу потрібного запису можна обчислювати, як це було розглянуто вище. Отже, в нашому прикладі довжина запису має бути рівною 32 - 12 + 8 = 28 двійкових розрядів, тобто з округленням до цілого числа байт – 4 байт. Таким чином, для кожного процесу, що виконується в комп'ютері, ОС створює таблицю сторінок розміром $4*Nв$ байт $= 4*2^{20} = 4$ Мб (рис. 10.10).

Іншим важливим чинником, що впливає на продуктивність системи, є частота сторінкових переривань, на яку, у свою чергу, впливають розмір сторінки і прийняті в цій системі правила вибору сторінок для вивантаження і завантаження. При неправильно вибраній стратегії заміщення сторінок можуть виникати ситуації, коли система витрачає велику частину часу даремно, на підкачування сторінок з оперативної пам'яті на диск і назад.

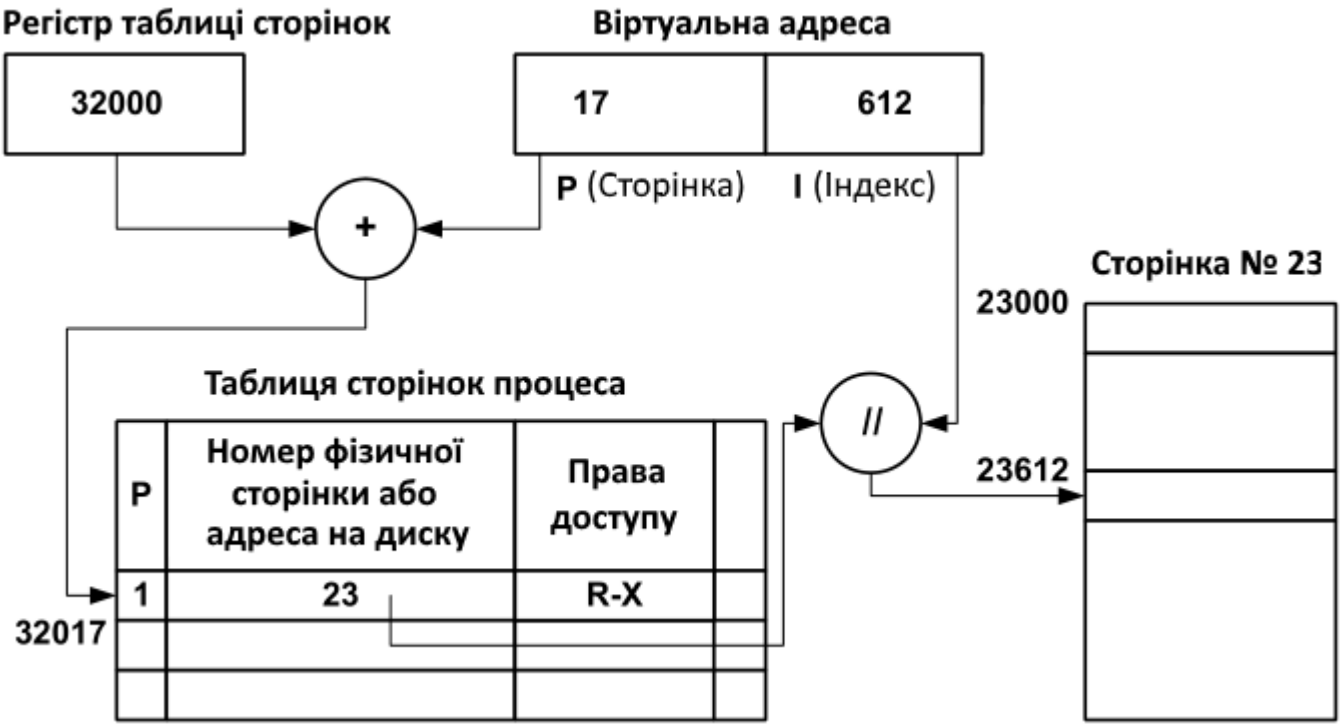

Рисунок 10.10 – Приклад розміщення сторінки в пам'яті

Щоб зменшити частоту сторінкових переривань, слід було б збільшувати розмір сторінки. Крім того, збільшення розміру сторінки зменшує розмір таблиці сторінок, тобто зменшує витрати пам'яті.

З іншого боку, якщо сторінка велика, то велика і фіктивна область в останній віртуальній сторінці кожної програми. В середньому на кожній програмі втрачається половина об'єму сторінки, що в сумі при великій сторінці може скласти істотну величину. З наведених міркувань виходить, що вибір розміру сторінки є складною оптимізаційною задачею, що вимагає обліку багатьох чинників.

На практиці ж розробники ОС і процесорів обмежуються деяким раціональним рішенням, придатним для широкого класу обчислювальних систем. Типовий розмір сторінки складає декілька кілобайт. Наприклад, найпоширеніші процесори x86 і Pentium компанії Intel, а також операційні системи, що встановлюються на цих процесорах, підтримують сторінки розміром 4096 байт (4 Кб). Так, процесор Pentium дозволяє використати також сторінки розміром до 4 Мб одночасно із сторінками об'ємом 4 Кб.

Час перетворення віртуальної адреси у фізичну значною мірою визначається часом доступу до таблиці сторінок. У зв'язку з цим таблицю сторінок прагнуть

розміщати в "швидких" запам'ятовуючих пристроях. Це може бути, наприклад, набір спеціальних регістрів або пам'ять, що використовує для зменшення часу доступу: асоціативний пошук і кешування даних.

Сторінковий розподіл пам'яті може бути реалізований в спрощеному варіанті, без вивантаження сторінок на диск. В цьому випадку усі віртуальні сторінки усіх процесів постійно знаходяться в оперативній пам'яті. Такий варіант сторінкової організації хоча і не надає користувачеві переваг роботи з віртуальною пам'яттю великого об'єму, але зберігає інші переваги сторінкової організації – дозволяє успішно боротися з фрагментацією фізичної пам'яті.

Дійсно, по-перше, програму можна розбити на частини і завантажити їх в розрізнені ділянки вільної пам'яті, а, по-друге, при завантаженні віртуальних сторінок ніколи не утворюється невикористаних залишків, оскільки розміри віртуальних і фізичних сторінок співпадають. Такий режим роботи системи управління пам'яттю використовується в деяких спеціалізованих ОС, коли потрібно забезпечити високу реактивність системи і здатність виконувати змінний набір додатків (приклад – ОС сімейства Novell NetWare 3.x і 4.x).

Нині відомі ще декілька методів підвищення ефективності функціонування сторінкової віртуальної пам'яті. До них належать:

- складніша структуризація віртуального адресного простору, наприклад, дворівнева (типова для 32-бітової адресації);
- використання спеціального високошвидкісного кеша для зберігання частини записів таблиці сторінок, який називають буфером швидкого перетворення адреси, або буфером пошуку трансляції (translation lookaside buffer – TLB);
- вибір оптимального розміру сторінки віртуальної пам'яті. Зупинимося на можливостях реалізації цих методів.

10.3.2 Багаторівневі таблиці сторінок

При сторінковому розподілі пам'яті розмір сторінки впливає на кількість записів в таблицях сторінок. Чим менші сторінки, тим об'ємнішими є таблиці сторінок процесів і тим більше місця вони займають в пам'яті. Оскільки в сучасних 32-розрядних процесорах максимальний об'єм віртуального адресного простору процесу може досягати до 4 Гб (2^{32}), то при розмірі сторінки 4 Кб (2^{12}) і довжині запису 4 байти для зберігання таблиці сторінок може знадобитися 4 Мб пам'яті.

Виходом в такій ситуації є зберігання в пам'яті тільки тієї частини таблиці сторінок, яка активно використовується в цей період часу. Оскільки сама таблиця сторінок зберігається в таких же сторінках фізичної пам'яті, що і описувані нею сторінки, то принципово можливо тимчасово витісняти частину таблиці сторінок з оперативної пам'яті. Це означає, що самі таблиці сторінок стають об'єктами сторінкової організації, як і будь-які інші сторінки.

Щоб обійти проблему необхідності постійного зберігання в пам'яті величезних таблиць сторінок деякі процесори використовують дворівневу таблицю сторінок. При такій схемі є **каталог таблиць сторінок (розділів)**, в якому кожен запис вказує на таблицю сторінок. Таким чином, якщо розмір каталогу – X, а максимальний розмір таблиці – Y, то процес може складатися максимум з X*Y сторінок. Максимальний розмір таблиці сторінок визначається умовою її розміщення в одній сторінці (як в Pentium).

На рис. 10.11 наведений приклад дворівневої схеми, типової для 32-бітової адресації. Подібна схема дозволяє істотно зберегти розмір призначеної для користувача таблиці сторінок, що розміщується в основній пам'яті (з 4 Мб до 4 Кб). Усі сторінки мають однаковий розмір, а розділи містять однакову кількість сторінок. Якщо розмір сторінки і кількість сторінок у розділі вибрати рівними мірі двійки (2^k і 2^n відповідно), то приналежність віртуальної адреси до розділу і сторінки, можна визначити дуже просто. Молодші k двійкових розрядів дають зміщення, наступні n розрядів є номером віртуальної сторінки, а старші розряди (позначимо їх кількість m), що залишилися, містять номер розділу.

Для кожного розділу будується власна таблиця сторінок. Кількість дескрипторів в таблиці і їх розмір підбираються такими, щоб об'єм таблиці виявився

рівним об'єму сторінки. Наприклад, у процесорі Pentium при розмірі сторінки 4 Кб довжина дескриптора сторінки складає 4 байти, а кількість записів в таблиці сторінок, що поміщається на сторінку, дорівнює 1024.

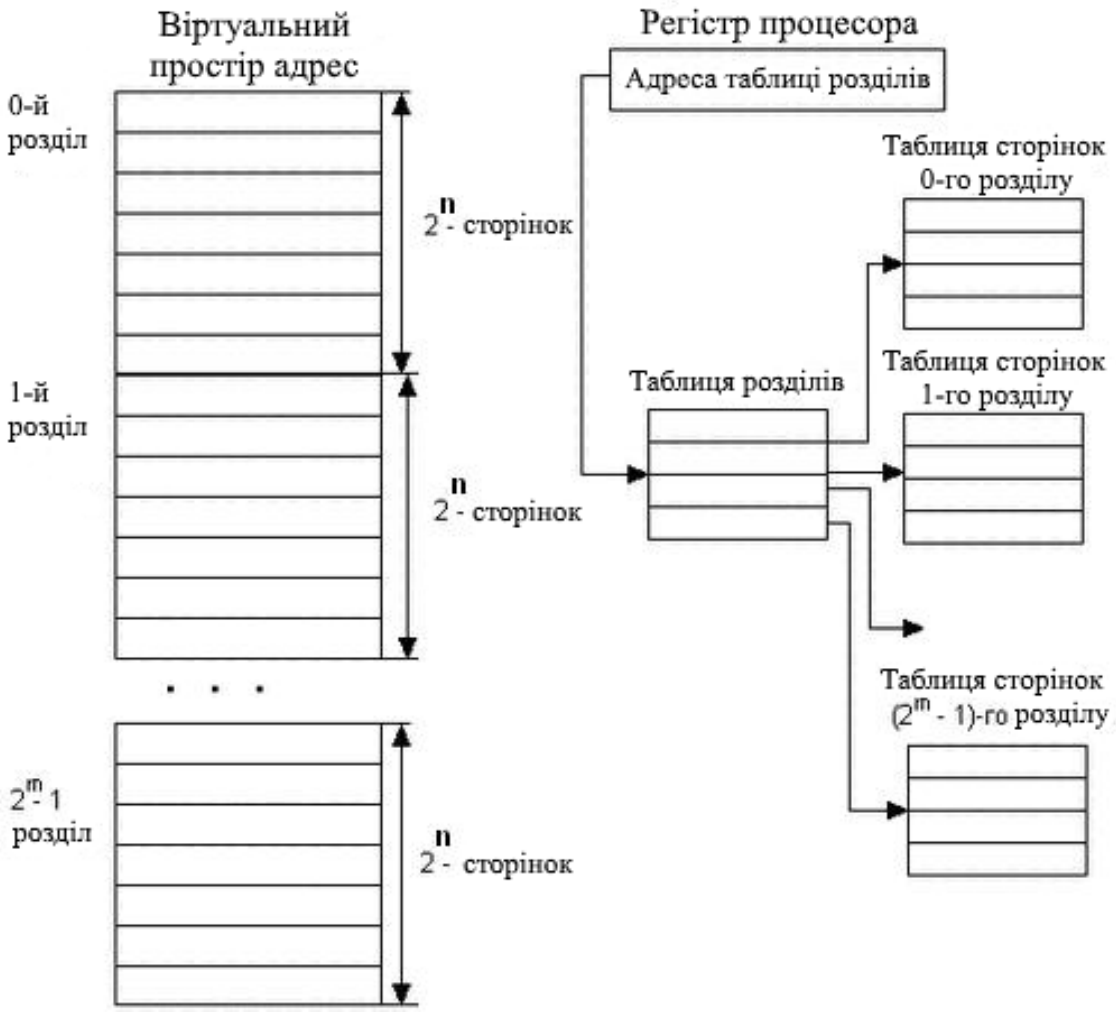

Рисунок 10.11 – Структура віртуального адресного простору з розділами

Ці дескриптори зведені в таблицю розділів (каталог сторінок). Фізична адреса таблиці розділів активного процесу міститься в спеціальному регістрі процесора і тому завжди відома операційній системі. Сторінка, що містить таблицю розділів (коренева таблиця), ніколи не вивантажується з основної пам'яті, інакше робота віртуальної пам'яті була б неможлива.

Вивантаження сторінок з таблицями сторінок дозволяє заощадити пам'ять, але при цьому призводить до додаткових часових витрат при отриманні фізичної адреси. Дійсно, може статися так, що та таблиця сторінок, яка містить потрібний дескриптор, в даний момент вивантажена на диск, тоді процес перетворення адреси призупиняється до тих пір, поки необхідна сторінка не буде знову завантажена в

пам'ять. Простежимо детальніше схему перетворення адрес для випадку дворівневої структуризації віртуального адресного простору (рис. 10.12):

1. Шляхом відкидання k+n молодших розрядів у віртуальній адресі визначається номер розділу, до якого належить ця віртуальна адреса.

2. По цьому номеру з таблиці розділів витягається дескриптор відповідної таблиці сторінок. Перевіряється, чи знаходиться ця таблиця сторінок в пам'яті. Якщо ні, відбувається сторінкове переривання і система завантажує потрібну сторінку з диска.

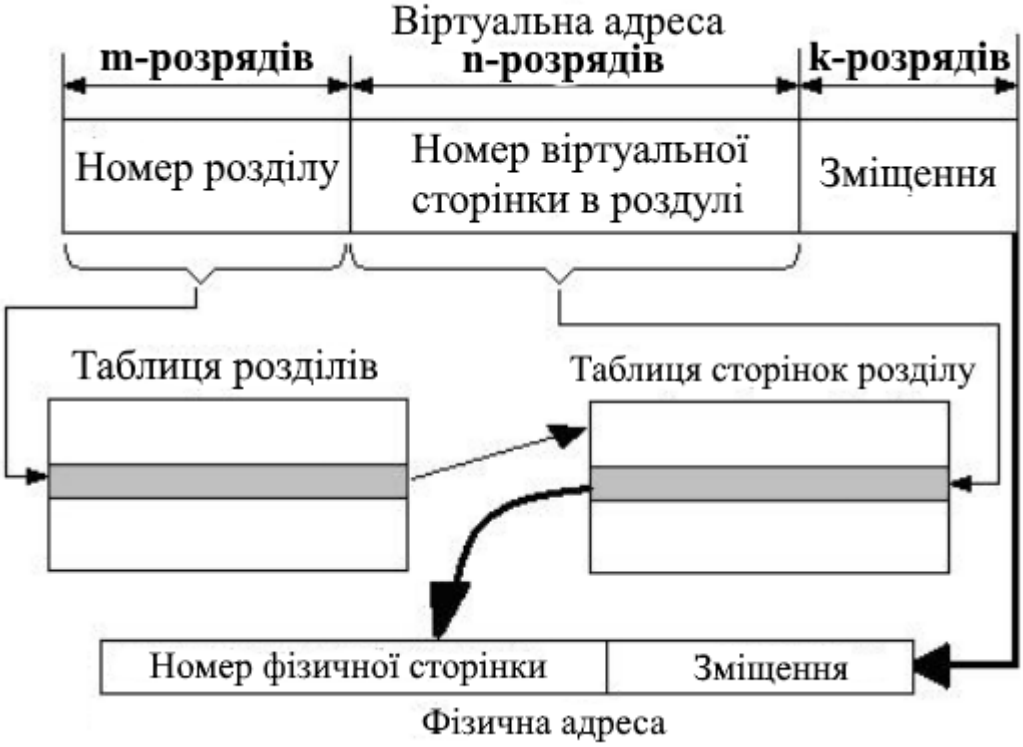

Рисунок 10.12 – Схема перетворення віртуальної адреси для дворівневої структуризації адресного простору

3. Далі з цієї таблиці сторінок витягається дескриптор віртуальної сторінки, номер якої міститься в середніх n розрядах перетворюваної віртуальної адреси. Знову виконується перевірка наявності цієї сторінки в пам'яті і при необхідності її завантаження.

4. З дескриптора таблиці сторінок визначається номер (базова адреса) фізичної сторінки, в яку завантажена ця віртуальна сторінка. До номера фізичної сторінки пристиковується зміщення, взяте з k молодших розрядів віртуальної адреси. У результаті виходить шукана фізична адреса.

На рис. 10.13 наведений приклад дворівневої схеми, типової для 32-бітової адресації. Подібна схема дозволяє істотно зберегти розмір таблиці сторінок користувача, що розміщується в основній пам'яті. В даному випадку віртуальний адресний простір для процесу користувача може складати 2^{32} = 4 Гб. При об'ємі сторінки 2^{12} = 4 Кб в цьому просторі розміщується $2^{32}/2^{12} = 2^{20}$ сторінок. Таким чином, таблиця сторінок користувача матиме 2^{20} 4-байтних записів загальним об'ємом 4 Мб.

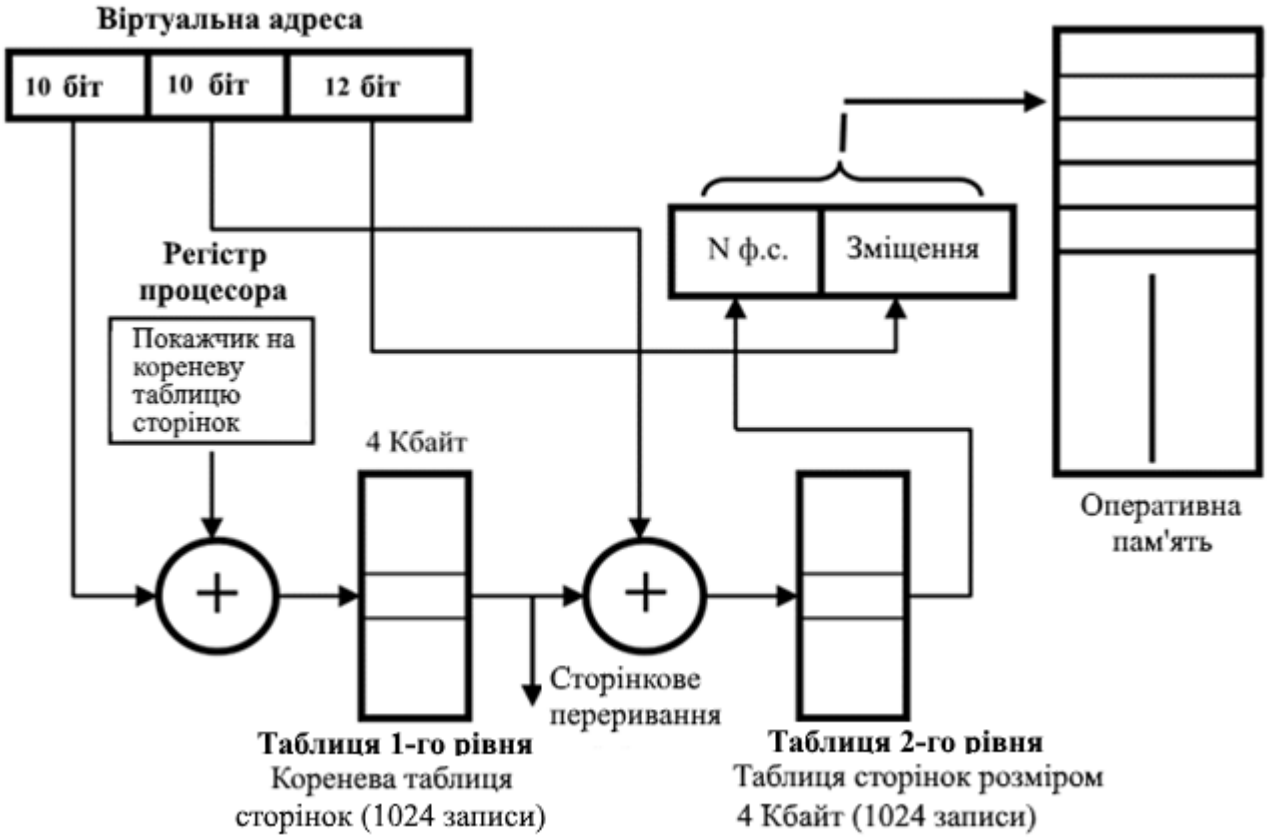

Рисунок 10.13 – Дворівнева схема таблиць сторінок

В основній пам'яті постійно знаходиться коренева таблиця, що містить 1024 записів, які вказують на початкову адресу призначеної для користувача таблиці сторінок (її об'єм, як вказано вище, 4 Мб). Вказівка на початкову адресу кореневої таблиці (активного процесу) заноситься в регістр процесора. Перші 10 біт віртуальної адреси використовуються для індексації в кореневій таблиці при пошуку записів про сторінку таблиці.

Якщо сторінка знаходиться в ОП, то наступні 12 біт віртуальної адреси використовуються для задання зміщення у фізичній сторінці ОП. Інакше генерується сторінкове переривання, але вже через відсутність потрібної сторінки процесу в ОП.

Таким чином, дворівнева схема, скорочуючи об'єм пам'яті для зберігання таблиці сторінок, в загальному випадку уповільнює перетворення віртуальної адреси із-за більшого числа можливих сторінкових переривань. Навіть якщо немає сторінкового переривання, потрібно три звернення до ОП замість двох при однорівневій сторінковій організації.

10.3.3 Буфери швидкого перетворення адреси (TLB)

Як уже відзначалося, проста схема сторінкової віртуальної пам'яті, по суті, подвоює час звернення до пам'яті: одне звернення для вибірки відповідного запису з таблиці сторінок, і ще одне звернення до адресних даних. А у разі використання дворівневих таблиць сторінок потрібні три операції доступу: до каталогу сторінок, до таблиці сторінок і безпосередньо за фізичною адресою. Для подолання цієї проблеми більшість схем віртуальної пам'яті використовують спеціальний високошвидкісний кеш для записів таблиці сторінок.

Практика використання віртуальної пам'яті показала, що для неї справедливий закон локалізації більшості звернень в невелику кількість нещодавно використаних сторінок (правило 90% до 10%), тому активно використовується тільки невелика частина таблиці сторінок.

Природне рішення проблеми прискорення – забезпечити комп'ютер апаратним пристроєм для відображення частини віртуальних сторінок у фізичні без звернення до таблиці сторінок. Тобто, мати невелику, швидку кеш-пам'ять, що зберігає необхідну на даний момент частину таблиці сторінок.

Модель локальності – принцип, покладений в роботу кеша. Якби доступ до будь-яких типів даних був випадковим, кеш був би даремний.

Для вирішення цієї проблеми більшість схем віртуальної пам'яті використовують спеціальний високошвидкісний кеш для записів таблиць сторінок, який називають **буфером швидкого перетворення адреси**, або **буфером пошуку трансляції (translation lookaside buffer – TLB)**, або буфером асоціативної трансляції, або іноді асоціативною пам'яттю.

Ґрунтуючись на правилі «дев'яносто до десяти» (правилі локалізації), що більшість програм схильна робити величезну кількість звернень до невеликої

кількості сторінок, комп'ютер забезпечується невеликим апаратним пристроєм, що служить для відображення віртуальних адрес у фізичні без проходу по таблиці сторінок. Цей пристрій знаходиться усередині диспетчера пам'яті і складається з декількох елементів, від 8 до 4096. Так, в архітектурі Intel-32 таких елементів до Pentium-4 було 32 (що забезпечувало 98% попадань в кеш), починаючи з Pentium-4 – 128.

Оскільки **асоціативна пам'ять** містить тільки деякі із записів таблиці сторінок, то кожен запис в TLB повинен включати поле з номером віртуальної сторінки. Окрім цього, кожен запис таблиці асоціативної пам'яті містить інформацію про одну віртуальну сторінку, а саме: біт зміни сторінки, код захисту (читання/запис/виконання), біт дійсності запису (чи використовується вона в даний момент) і номер фізичного сторінкового блоку.

Пам'ять називається **асоціативною**, оскільки в ній відбувається одночасне порівняння номера віртуальної сторінки, що відображається, з відповідним полем в усіх рядках цієї невеликої таблиці.

Коли віртуальна адреса представляється диспетчером пам'яті для відображення, апаратура спочатку переконується в тому, що номер його віртуальної сторінки є присутнім у буфері TLB шляхом порівняння адреси з усіма записами одночасно. Якщо знайдено співпадіння, то сторінковий блок береться прямо з буфера, без переходу до таблиці сторінок (рис. 10.14).

Якщо номер віртуальної сторінки не знаходиться в TLB, то диспетчер пам'яті виконує звичайний пошук в таблиці сторінок. Потім він видаляє один із записів з буфера (зазвичай старіший елемент) і замінює його тільки що знайденим записом з таблиці сторінок, з ймовірністю того, що цей запис знову незабаром зажадається. Тут ми стикаємося з традиційною для будь-якого кеша проблемою заміщення, а саме який із записів в кеші необхідно змінити. Конструкція асоціативної пам'яті повинна організовувати записи так, щоб можна було прийняти рішення про те, який із старих записів має бути видалений при внесенні нових.

Основні принципи, за якими організується робота асоціативної пам'яті, показані на рис. 10.15. Процесор апаратно здатний одночасно опитувати всі записи TLB для визначення того, яка з них відповідає заданому номеру сторінки. Такий підхід

відомий як асоціативне відображення (associative mapping), на відміну від прямого відображення, або індексування, що застосовується для пошуку в таблиці сторінок.

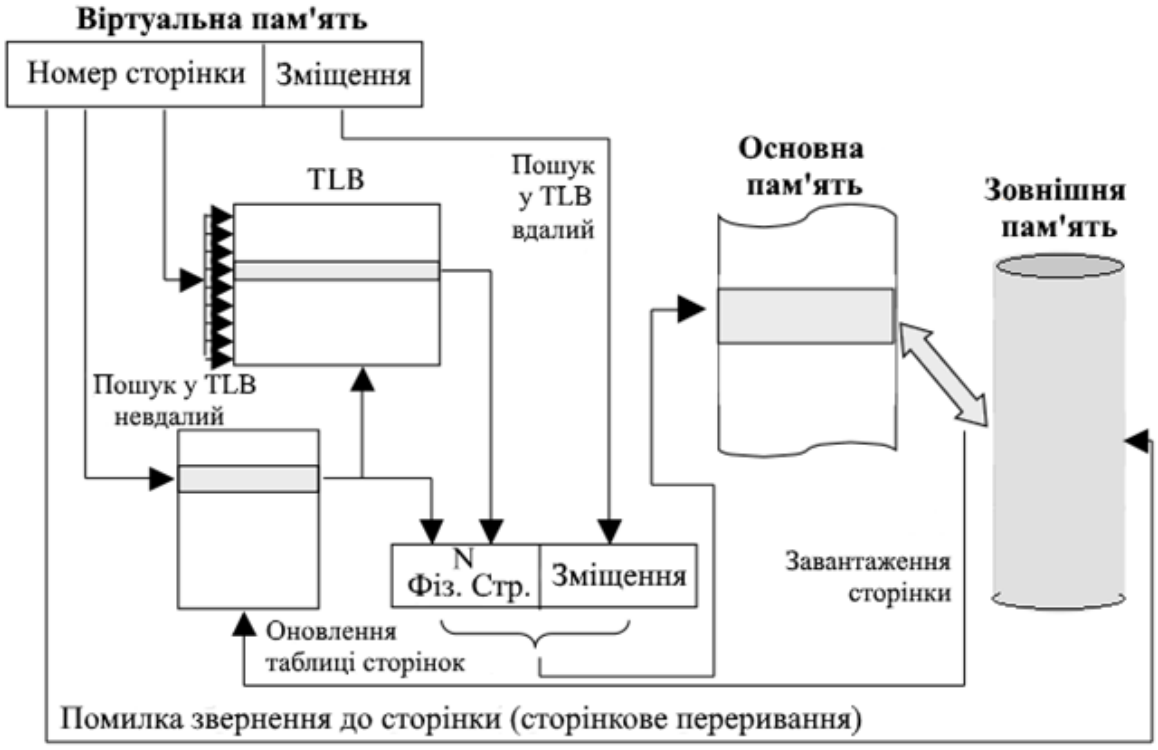

Рисунок 10.14 – Буфер швидкої переадресації

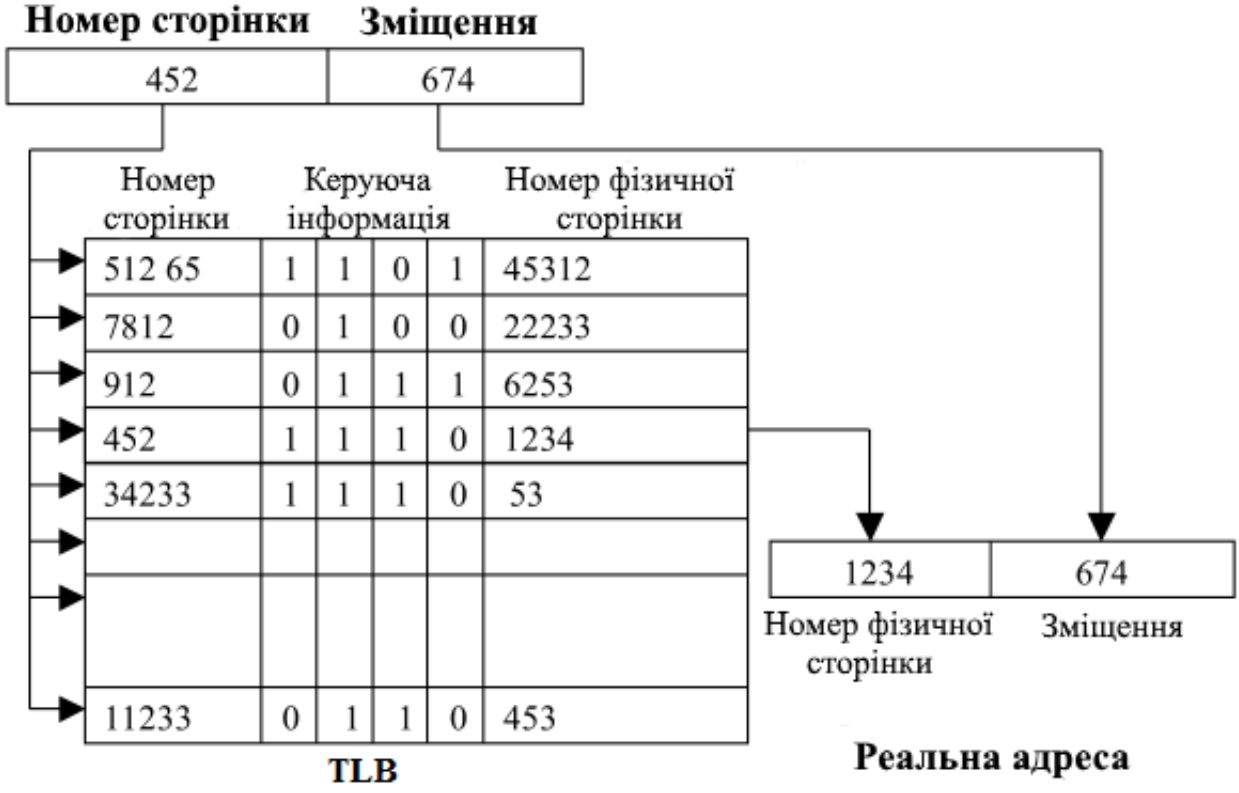

Рисунок 10.15. Асоціативна пам'ять

Число вдалих пошуків номера сторінки в асоціативній пам'яті по відношенню до загального числа пошуків називається **hit** (співпадіння) **ratio** (пропорція, відношення). Іноді також використовується термін «відсоток попадань в кеш». Звернення до одних і тих же сторінок підвищує **hit ratio**. Чим більше **hit ratio**, тим менше середній час доступу до даних, що знаходяться в оперативній пам'яті.

Припустимо, наприклад, що для визначення адреси в разі кеш-промаху через таблицю сторінок потрібно 100 нс (t1), а для визначення адреси в разі кеш- попадання через асоціативну пам'ять – 20 нс(t2). З 90% hit ratio (p – ймовірність кеш-попадания) середній час визначення адреси за формулою повної ймовірності:

$$t = t1 + t2*p = 100*0,1 + 20*0,9 = 28 \text{ нс}.$$

У разі перемикання контексту в архітектурі Intel-32 необхідно очистити увесь кеш, оскільки для кожного процесу є своя таблиця сторінок, і ті ж самі номери сторінок для різних процесів можуть відповідати різним фреймам у фізичній пам'яті. Таким чином, використання асоціативної пам'яті збільшує час перемикання контексту. Дворівнева (асоціативна пам'ять + таблиця сторінок) схема перетворення адреси є яскравим прикладом ієрархії пам'яті, заснованої на використанні принципу локальності, про що ми говоритимемо в наступному розділі.

Слід підкреслити, що механізм віртуальної пам'яті повинен взаємодіяти з кешем оперативної пам'яті (окрім TLB). Ця взаємодія показана на рис. 10.16 [10].

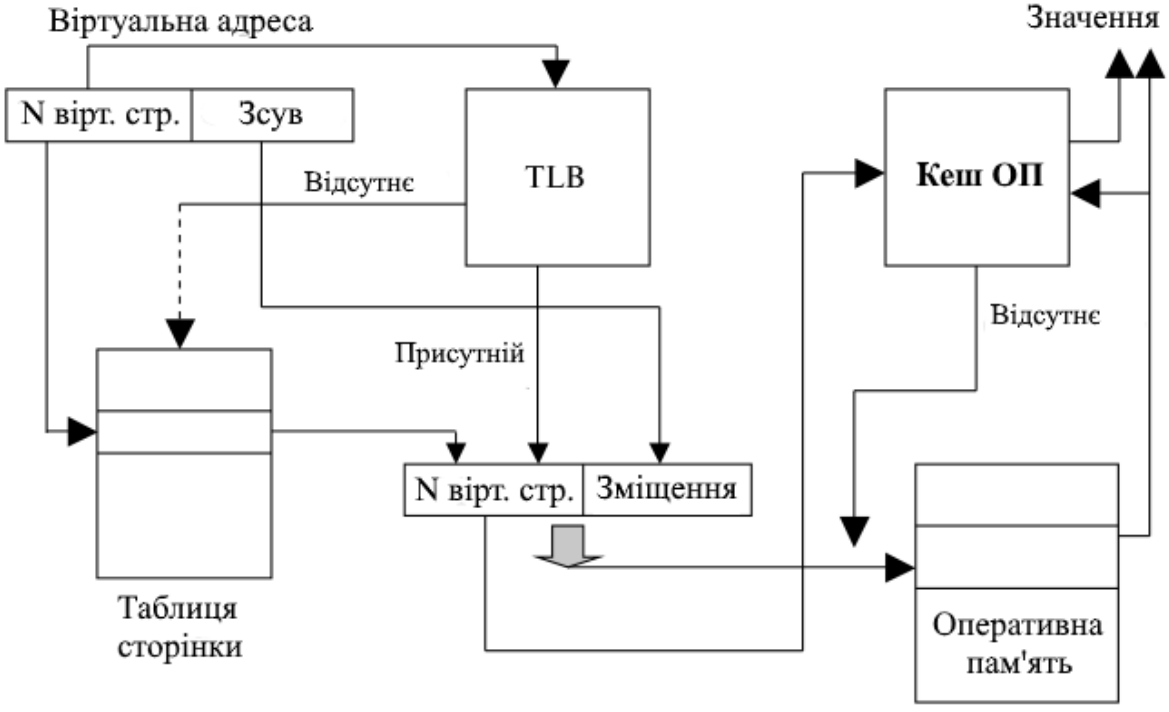

Рисунок 10.16. Використання кешу оперативної пам'яті

Спочатку відбувається звернення до TLB для з'ясування наявності в ньому відповідного запису таблиці сторінок. При позитивному результаті шляхом об'єднання номера фізичної сторінки, що отримується з TLB, і зміщення генерується фізична адреса. Якщо необхідного запису в TLB немає, вона вибирається з таблиці сторінок. Після отримання фізичної адреси в обох ситуаціях виконується звернення до кеша оперативної пам'яті для з'ясування наявності в ньому блоку з необхідною фізичною адресою. Якщо відповідь позитивна, то необхідне значення (код або дані) передається процесору. Інакше робиться вибірка слова з основної пам'яті і оновлюється вміст кеша основної пам'яті.

10.3.4 Інвертовані таблиці сторінок, хеш-таблиці

Традиційні таблиці сторінок вимагають по одному запису на кожну сторінку. Якщо адресний простір складається з 2^{32} байт з розміром сторінки 4096 байт(2^{12}), тоді таблиця сторінок містить більше мільйона записів і займатиме мінімум 4 Мб (2^{20}). Зрозуміло, що виділяти таку кількість оперативної пам'яті під таблиці сторінок недоцільно. А при 64-розрядному адресному просторі з розміром сторінки 4 Мб і розміром запису таблиці в 8 байт, таблиця займе більше 30 Тб (біля 4 триліонів записів). Це нереально навіть у майбутньому.

Тому ще одним підходом до використання одно- або дворівневих таблиць сторінок являється застосування **інвертованої таблиці сторінок** (рис. 10.17). Цей підхід застосовується на машинах PowerPC, деяких робочих станціях Hewlett-Packard, IBM RT, IBM AS/400 і ряді інших. Особливо інвертовані таблиці сторінок знайшли широке застосування на 64-розрядних машинах.

У цій моделі таблиця містить по одному запису на кожен сторінковий кадр фізичної пам'яті, а не на сторінку у віртуальному адресному просторі. Істотно, що достатньо однієї таблиці для усіх процесів. Наприклад, при 64-розрядних віртуальних адресах, при розмірі сторінок 4 Кб і 1 Гб оперативної пам'яті інвертована таблиця сторінок зажадає усього лише 262144 записів. Кожен запис інвертованої таблиці сторінок містить номер процесу і номер віртуальної сторінки. Слід зауважити, що інвертовані сторінкові таблиці не зберігають інформації про розміщення

нерезидентних сторінок на вторинних пристроях зберігання. Ця інформація підтримується ОС.

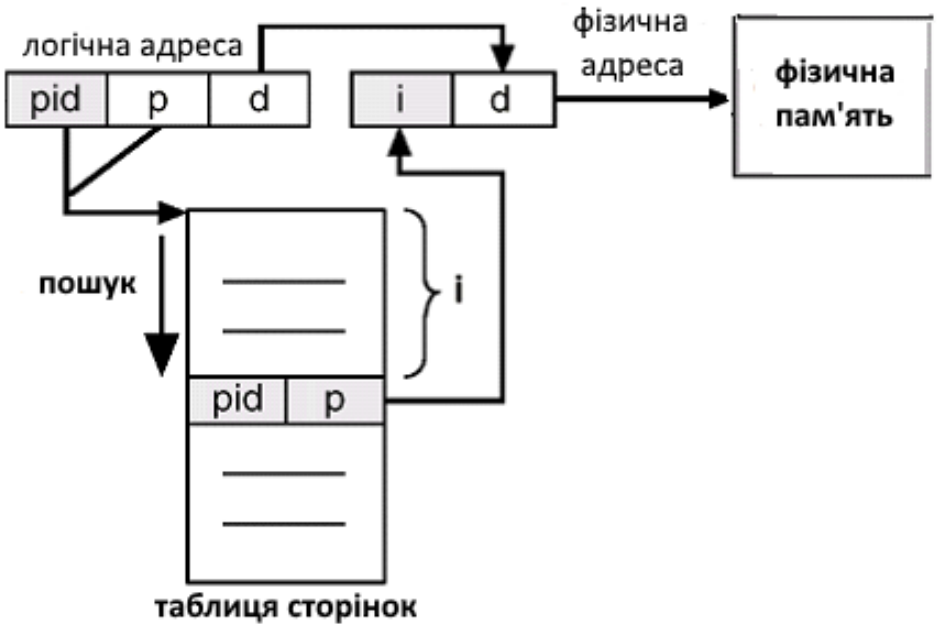

Рисунок 10.17 – Інвертована таблиця сторінок

Хоча інвертовані таблиці сторінок економлять значну кількість місця, вони мають серйозний недолік – записи в них (як і в асоціативній пам'яті) не відсортовані за збільшенням номерів віртуальних сторінок, що ускладнює трансляцію адреси. Коли процес N звертається до віртуальної сторінки P, апаратне забезпечення не може більше знайти фізичну сторінку, використовуючи номер P як індекс в таблиці сторінок. Замість цього здійснюється пошук запису (**pid, P**) в усій інвертованій таблиці сторінок, де **pid** – ідентифікатор процесу N. Вийти з цього положення можна, використовуючи розглянутий раніше буфер швидкого перетворення адреси (TLB). Але при невдалому пошуку в буфері TLB пошук в інвертованій таблиці сторінок повинен виконуватися програмно. Один з можливих способів удосконалити його – підтримувати хеш-таблицю віртуальних адрес.

При такому підході частина віртуальної адреси, яка представляє номер сторінки, відображається в хеш-таблицю з використанням простої функції хешування. Наприклад, для N елементів, які зберігаються в таблиці розміром M≥N (причому M не набагато більше N), мітка елементу перетвориться на майже випадкове число n між 0 і M-1 методом ділення мітки по модулю M.

Цілочисельна функція **hash** визначає значення s з відрізку [0, M - 1]. При пошуку елемента s спочатку обчислюється hash(s), а потім виконується пошук тільки в списку Shash(s). Оскільки на один і той же запис хеш-таблиці можуть відображатися декілька віртуальних адрес, то для обробки переповнення використовується технологія ланцюжків, які на практиці досить короткі – як правило, від одного до двох записів (рис. 10.18).

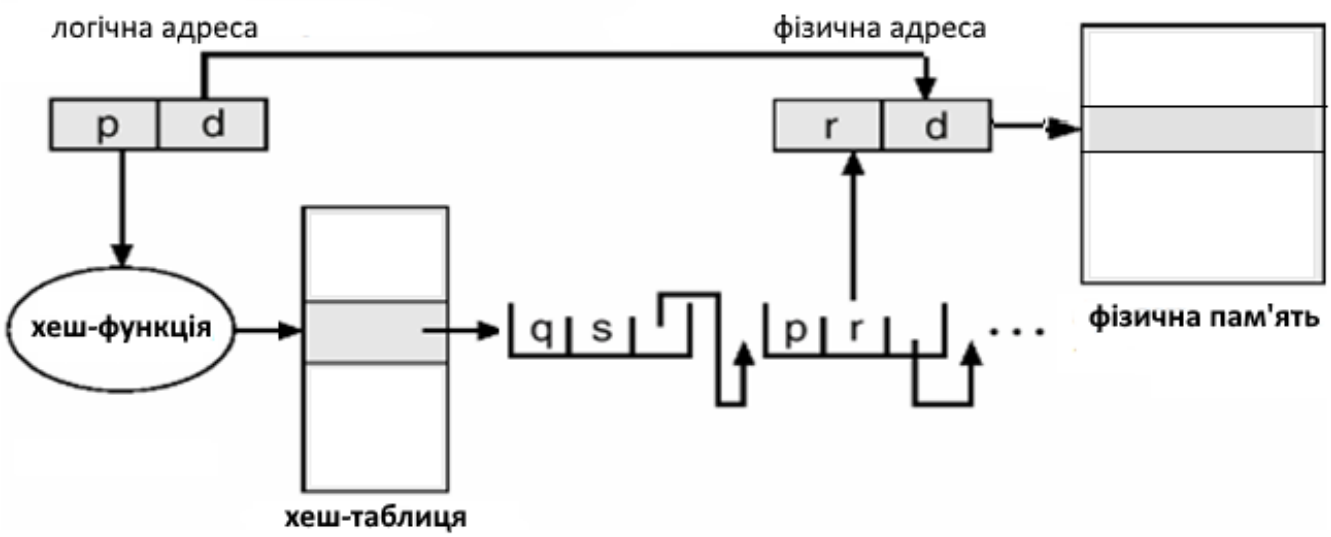

Рисунок 10.18 – Хеш-таблиця сторінок

Хеш-таблиця містить покажчик на інвертовану таблицю сторінок. Кожній сторінці реальної пам'яті при цьому відповідає один запис в хеш-таблиці і в інвертованій таблиці сторінок. Таким чином, для зберігання таблиць потрібна фіксована частина основної пам'яті, незалежно від розміру і кількості процесів і підтримуваних віртуальних сторінок.

Наприклад, вставка елементу в таблицю відбувається таким чином.

1. Перетворимо мітку елементу в майже випадкове число s між 0 і M-1 методом ділення мітки по модулю M.
2. Використовуємо отримане значення s як індекс в хеш-таблиці:

А. Якщо відповідний запис в таблиці порожній, значить, елемент раніше не був збережений в таблиці.

Б. Якщо запис вже зайнятий і її мітка відповідає шуканій, значить, знайдений необхідний елемент.

В. Якщо запис зайнятий, і її мітка не відповідає заданій, продовжуємо пошук в області переповнювання.

Середня тривалість пошуку елемента при відкритій хеш-таблиці з використанням таблиць переповнювання з ланцюжками, рівна 1/2M, для великих значень N=M прагне до 1.5, зокрема для бінарного пошуку в сортованому списку тривалість пошуку елементу рівна Log_2M.

10.3.5 Визначення найкращого розміру сторінки

При виборі розміру сторінки треба враховувати декілька чинників. Один з них – внутрішня фрагментація, яка безпосередньо залежить від розміру сторінки. Внутрішня фрагментація зменшується зі зменшенням розміру сторінки. Проте чим менше сторінка, тим більше їх потрібно для процесу, що означає збільшення розміру таблиці сторінок. Розмір сторінки впливає і на частоту виникнення переривання через відсутність сторінки в основній пам'яті.

Реально розміри сторінок різних комп'ютерів складають такі значення: 512 байт (сімейство VAX, IBM AS/400), 4 Кб (IBM 370), 8 Кб (DEC Alpha), від 4 Кб до 4 Мб (Pentium).

Визначення найкращого розміру сторінки вимагає урівноваження декількох чинників. Тому не існує абсолютного оптимального рішення. Існує два аргументи на користь маленького розміру сторінок. Випадково вибраний текст, дані або сегмент стека не заповнюють цілу кількість сторінок. В середньому половина останньої сторінки виявляється порожньою. Якщо в пам'яті n сегментів при розмірі сторінки p байт, то np/2 байт буде витрачений даремно в результаті внутрішньої фрагментації. Це розумний аргумент на користь сторінок невеликого розміру.

Інший аргумент стає очевидним, якщо представити програму, що складається з восьми послідовних етапів, по 4 Кб кожен. При розмірі сторінки 32 Кб програмі мають бути постійно виділені 32 Кб. При розмірі сторінки 16 Кб їй потрібні тільки 16 Кб. При розмірі сторінки 4 Кб, або менше, програма вимагає усього лише 4 Кб у будь-який момент часу. Тобто, великий розмір сторінки швидше, чим маленький, стане причиною того, що в пам'яті знаходиться невживана частина сторінки.

З іншого боку, невеликий розмір сторінки означає, що програмам буде потрібно багато сторінок, отже потрібна величезна таблиця сторінок. Як правило, сторінка за раз переноситься на диск і з нього. При цьому велика частина часу йде на пошук

циліндра і затримку обертання. Так що переміщення маленької сторінки займає майже стільки ж часу, скільки і великої. Може знадобитися 64*10 мс, щоб завантажити 64 сторінки розміром 512 байт, і усього лише 4*12 мс для завантаження 4-х сторінок по 8 Кб.

Враховуючи усе це можна математично проаналізувати розмір сторінки. Нехай середній розмір процесу рівний S байт, а сторінки – P байт. Крім того, припустимо, що запис для кожної сторінки вимагає E байт. Тоді приблизна кількість сторінок, необхідна для процесу, рівна S/P, що займе SE/P байт для таблиці сторінок. Втрата пам'яті в останній сторінці процесу рівна P/2. Таким чином, загальні накладні витрати внаслідок підтримки таблиці сторінок і втрати від внутрішньої фрагментації дорівнюють сумі цих складових:

витрата = SE/P + P/2.

Перший доданок (розмір таблиці сторінок) збільшується при зменшенні розміру сторінок. Другий доданок (внутрішня фрагментація) при збільшенні розміру сторінок зростає. Оптимальний варіант повинен знаходитися десь посередині. Якщо узяти 1-у похідну по змінній P і прирівняти її до нуля, отримаємо рівність: - SE/P2 + ½ = 0. З цієї рівності можна отримати формулу, що дає оптимальний розмір сторінок (беручи до уваги тільки втрати пам'яті на фрагментацію і розмір таблиці сторінок). У результаті вийде: P = √2SE.

Для середнього розміру процесу S = 1 Мб і розміру запису в таблиці сторінок E = 8 байт оптимальний розмір сторінки дорівнюватиме 4 Кб. У сучасних комп'ютерах частіше зустрічаються розміри сторінок 4 Кб або 8 Кб. Оскільки пам'яті стає більше, то розмір сторінок також має тенденцію зростання (але залежність нелінійна).