## 10.5 СЕГМЕНТНО-СТОРІНКОВА ОРГАНІЗАЦІЯ ПАМ'ЯТІ. СВОПІНГ.

І сторінкова, і сегментна організація мають свої переваги. Сторінкова організація усуває зовнішню фрагментацію, і тим самим забезпечує ефективне використання основної пам'яті. Крім того, оскільки переміщувані блоки мають фіксований, однаковий розмір, то полегшується створення ефективних алгоритмів управління пам'яттю.

Як видно з назви, цей метод є комбінацією сторінкового і сегментного розподілу пам'яті і, внаслідок цього, поєднує в собі переваги обох підходів.

При великому розмірі сегментів може стати незручним або навіть неможливим зберігання їх цілком в оперативній пам'яті. Це наштовхує на ідею застосування до них сторінкової організації, щоб мати справу тільки з тими сторінками, які потрібні в даний момент. Підтримка сторінкових сегментів реалізована в декількох системах.

### 10.5.1 Загальна схема організації сегментно-сторінкової пам'яті

Уперше сегментація зі сторінковою організацією пам'яті була застосована в ОС MULTICS (1965 р.) [12]. Система MULTICS забезпечувала кожну програму віртуальною пам'яттю розміром аж до $2^{18}$ сегментів (більше 250000), кожен з яких міг бути розміром до 65536 36-розрядних слів завдовжки. Розробники системи MULTICS вирішили трактувати кожен сегмент як віртуальну пам'ять і розбити його на сторінки, комбінуючи переваги сторінкової організації пам'яті з перевагою сегментації.

У такій комбінованій системі адресний простір користувача розбивається на ряд сегментів на розсуд програміста. Кожен сегмент у свою чергу розбивається на сторінки фіксованого розміру, які рівні сторінці фізичної пам'яті. З точки зору програміста, логічна адреса в цьому випадку складається з номера сегменту і зміщення в ньому. З позиції операційної системи зміщення в сегменті слід розглядати як номер сторінки певного сегменту і зміщення в ній (рис. 10.25).

Приклад отримання фізичної адреси і витяг з пам'яті необхідного елементу для цього способу представлений на рис. 10.26.

Цей спосіб організації віртуальної пам'яті вносить ще більшу затримку доступу до пам'яті. Необхідно спочатку вичислити адресу дескриптора сегменту і прочитати її, потім вичислити адресу елементу таблиці сторінок цього сегменту і витягнути з

пам'яті необхідний елемент, і вже тільки після цього можна до номера фізичної сторінки приписати номер комірки в сторінці (індекс). Щоб уникнути такої затримки, вводиться кешування, причому кеш, як правило, будується за асоціативним принципом.

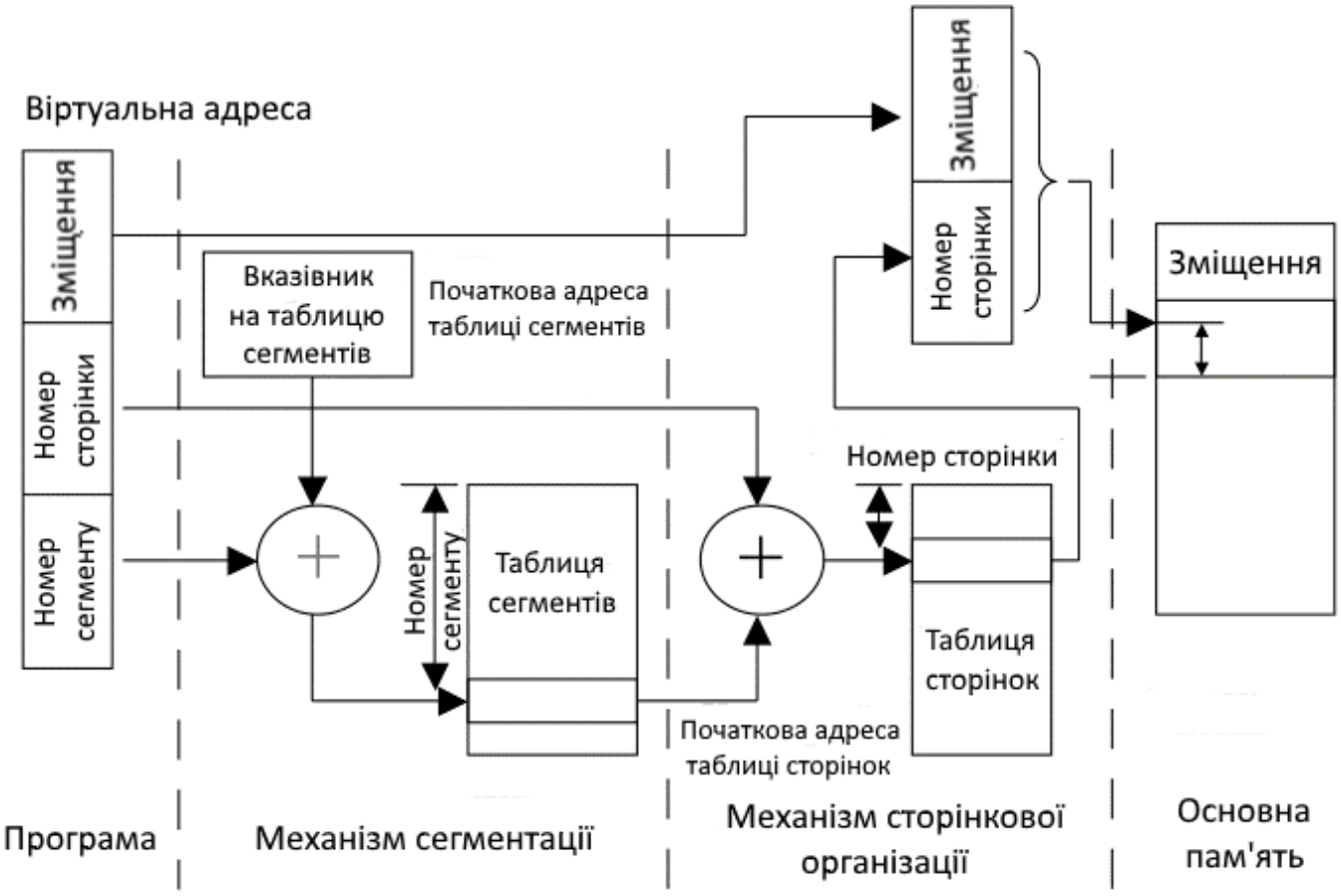


Рисунок 10.25 – Сегментно-сторінкова організація пам'яті

Апаратура системи MULTICS містила буфер швидкого перетворення адреси (TLB) розміром 16 слів, який був здатний здійснювати пошук паралельно по усіх своїх записах для заданого ключа. У буфері швидкого перетворення адреси зберігалися адреси 16 сторінок, до яких відбувалися самі останні звернення. Програми, в яких робочий набір був менше розміру TLB, зберігали адреси всього робочого набору в TLB, і отже, ці програми працювали ефективно, інакше відбувалася помилка TLB.

Сегментація зручна для реалізації захисту і спільного використання сегментів різними процесами. Оскільки кожен запис таблиці сегментів включає початкову адресу і значення довжини, програма не в змозі ненавмисно звернутися до основної пам'яті за межами сегменту. Для того щоб відрізнити сегменти від індивідуальних,

записи таблиці сегментів, що розділяються, містять 1-бітове поле, що має два значення: shared (розділяється) або private (індивідуальний). Для здійснення спільного використання сегменту він поміщається у віртуальний адресний простір декількох процесів, при цьому параметри відображення цього сегменту налаштовуються так, щоб вони відповідали одній і тій же області оперативної пам'яті.

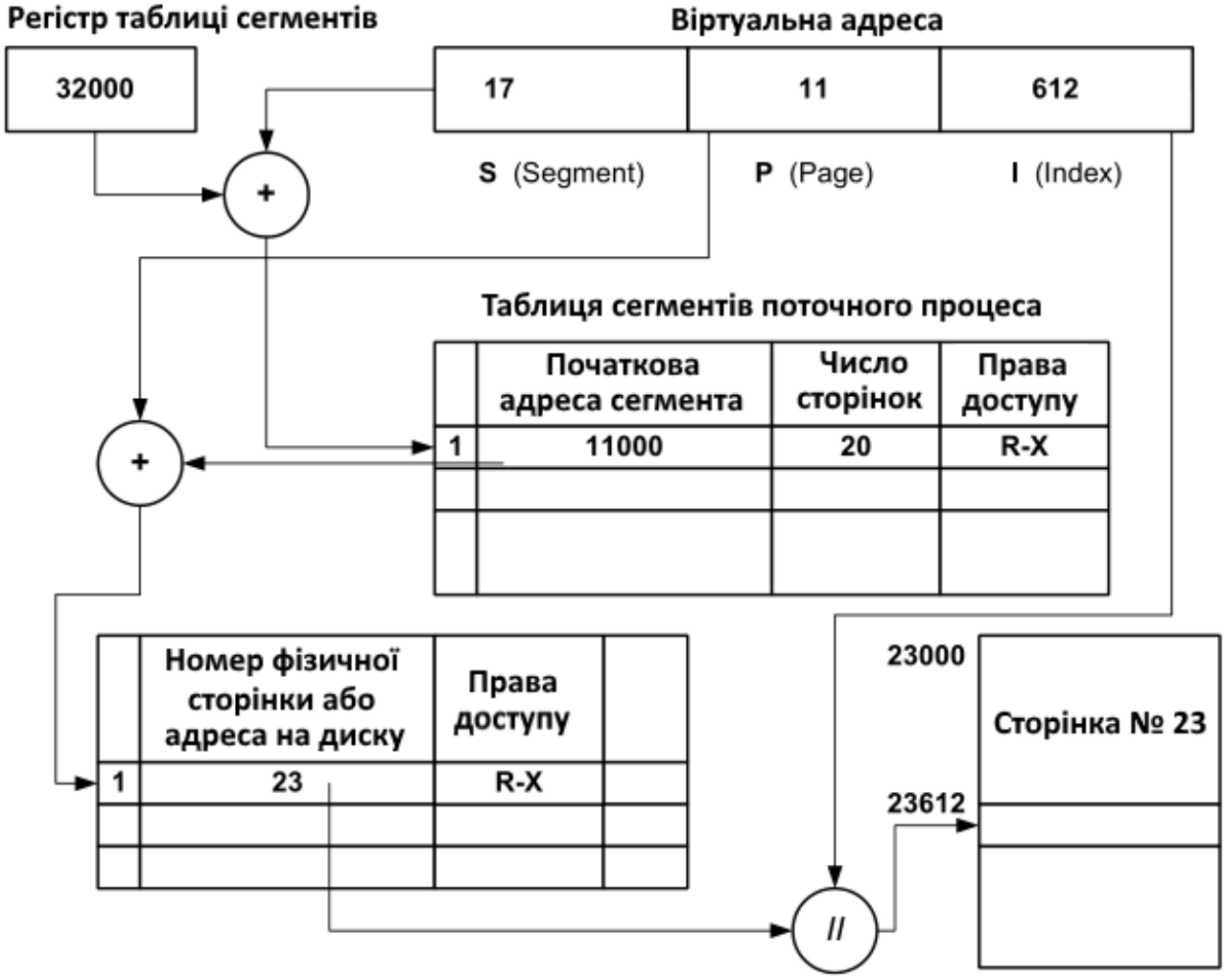


Рисунок 10.26 – Приклад отримання фізичної адреси при сегментно- сторінковій організації пам'яті

Незважаючи на переваги над другими методами розподілу пам'яті сегментно-сторінкове розподіл пам'яті вимагає дуже значних витрат обчислювальних ресурсів і його не так просто реалізувати. Використовується він нечасто, причому в дорогих, потужних обчислювальних системах. Можливість реалізувати сегментно-сторінковий розподіл пам'яті закладена і в сімейство мікропроцесорів i80x86, однак внаслідок слабкої апаратної підтримки, труднощів при створенні систем програмування та операційної системи, практично він не використовується в ПК [9].

### 10.5.2 Сегментно-сторінкова організація пам'яті в Windows

Розглянемо стисло сучаснішу організацію сегментно-сторінкової пам'яті, яка застосовується в системі Intel Pentium. Докладніше сегментно-сторінкова організація пам'яті ОС Windows розглядалася в підрозділі «4.6 Засоби підтримки механізмів віртуальної пам'яті».

Віртуальна пам'ять в системі Pentium схожа на пам'ять в системі MULTICS, включаючи наявність як сегментації, так і сторінкової організації. Але система MULTICS має 256 Кб незалежних сегментів, кожен до 64 Кб 36-розрядних слів, а система Pentium підтримує 16 Кб незалежних сегментів, кожен до 1 млрд 32-розрядних слів. Хоча система Pentium має менше сегментів, їх більший розмір куди важливіше, оскільки програми, яким потрібно більш ніж 1000 сегментів, зустрічаються досить не часто, тоді як багатьом програмам потрібні великі за розміром сегменти.

Основа віртуальної пам'яті системи Pentium складається з двох таблиць: **локальної таблиці дескрипторів – LDT** (Local Descriptor Table) і **глобальної таблиці дескрипторів – GDT** (Global Descriptor Table). У кожної програми є своя власна таблиця LDT, але глобальна таблиця дескрипторів, яку спільно використовують усі програми в комп'ютері, всього одна. У таблиці LDT описуються сегменти, локальні для кожної програми, включаючи код цих програм, їх дані, стек тощо, а в таблиці GDT описуються системні сегменти, включаючи саму операційну систему.

Щоб отримати доступ до сегменту, програма, що працює в системі Pentium, спочатку завантажує селектор для цього сегменту в один з шести сегментних регістрів машини. Під час виконання програми регістр CS містить селектор для сегменту коду, а регістр DS зберігає селектор для сегменту даних. Кожен селектор, як показано на рис. 10.27, є 16-розрядним цілим числом, що складається з трьох полів:

– індексу, який задає послідовний номер дескриптора в таблиці GDT або LDT (13 біт);

– покажчика типу використовуваної таблиці дескрипторів: GDT або LDT (1 біт);

– необхідного рівня привілеїв – RPL (2 біти), це поле використовується механізмом захисту даних (0 – ядро, 1 – системні вузли, 2 – бібліотеки спільного доступу, 3 – призначені для користувача програми). Рівні привілейованості забороняють виконуваному коду звернутися до нижчого рівня.

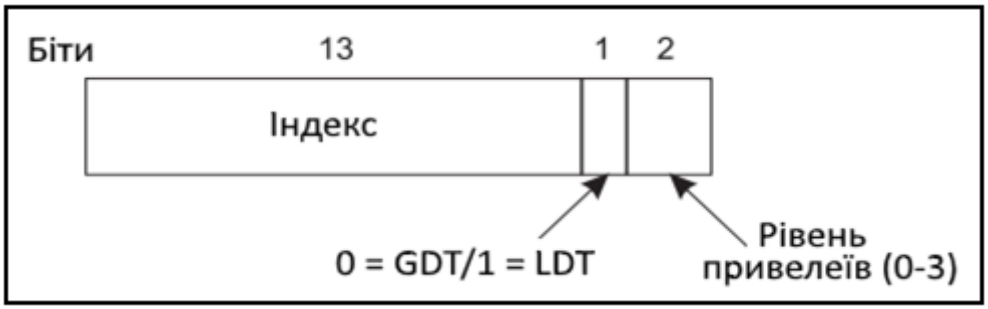


Рисунок. 10.27 – Формат селектора в Pentium

Дескриптор 0 заборонений. Його можна без всякого побоювання завантажити в сегментний регістр, щоб позначити, що цей сегментний регістр в даний момент недоступний. Спроба ним скористатися призведе до системного переривання. Щоб полегшити визначення місця розташування дескриптора, був майстерно підібраний формат селектора. Спочатку на основі біта 2 селектора вибирається локальна або глобальна таблиця дескрипторів. Потім селектор копіюється у внутрішній робочий регістр, і значення трьох молодших бітів встановлюються в 0. Нарешті, до цієї копії додається адреса однієї з таблиць, LDT або GDT, щоб отримати прямий покажчик на дескриптор. Наприклад, селектор 72 посилається на запис 9 (72/8) в глобальній таблиці дескрипторів, яка розташована за адресою в таблиці GDT+72.

Під час завантаження селектора в сегментний регістр, з локальної або глобальної таблиці дескрипторів витягається відповідний дескриптор, який, щоб прискорити звернення до нього, зберігається в мікропрограмних регістрах. Як показано на рис. 10.28, дескриптор сегмента складається з 8 байтів, в які входить базова адреса сегменту (32 біта), 20-бітний розмір сегмента в елементах (див. G) та інша інформація.

| 31<---------- 32 біта ----------->0 | | | | | | | | | | | | Адрес |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Base 15-0 | | | | | | Lim 15-0 | | | | | | 0 |
| Base 31-24 | G | D | X | U | Lim19-16 | P | DPL | S | Type | A | Base 23-16 | 4 |

Рисунок 10.28 – Формат дескриптора сегмента даних або коду в Pentium

Base – 32-бітова початкова адреса сегменту в лінійному фізичному адресному просторі.

Lim (Limit) – 20-бітовий розмір сегменту в елементах (см G) мінус 1.

G (Granularity) – біт гранулярності, якій задає величину сегментного елементу: 0 – байт, 1 – сторінка (4 Кб). Таким чином, максимальний розмір сегменту може складати 1 Mb байтів або 1 Mb сторінок (4 Gb).

P (Present) – біт присутності, що містить 1, якщо сегмент знаходиться у фізичній пам'яті.

DPL (Descriptor Privilege Level) – двубітне поле рівня привілеїв, що відповідає сегменту.

S (System) – ознака системного об'єкту (0 – системний).

Type – 3 біта, що визначають цільове використання сегменту (дані, код, стек, дозвіл на читання і запис).

A (Accessed) – біт доступу, що встановлюється при зверненні до сегменту. D (Default Size) – біт розміру за умовчанням, який задає розмір операндів в сегменті: 0 – 16 бітів, 1 – 32 біта; у сегменті коду це відповідає 16-бітовому коду процесора 80286 і 32-бітовому Pentium.

X – зарезервований біт.

U (User) – призначений для використання програмістами, процесор його ігнорує.

Як тільки мікропрограма дізнається, який сегментний регістр використовується, вона може знайти у своїх внутрішніх регістрах повний дескриптор, що відповідає цьому селектору. Якщо сегмент не існує (селектор дорівнює 0) або в даний момент вивантажений, виникає системне переривання.

Потім апаратура використовує поле межі (Розмір, Limit), щоб перевірити, чи не виходить зміщення за межу сегменту, і в цьому випадку також виникає системне переривання. Для надання розміру сегменту в дескрипторі має бути 32- розрядне поле, але доступні тільки 20 біт, тому використовується інша схема. Якщо поле G (Granularity – міра деталізації) дорівнює нулю, в полі Розмір (Limit) міститься точний розмір сегменту аж до 1 Мб. Якщо воно дорівнює 1, то в полі Розмір надається розмір сегменту в сторінках, а не в байтах. У системі Pentium використовується фіксований

розмір сторінок, рівний 4 Кб, тому 20 бітів достатні для сегментів розміром до 232 байтів.

Припустимо, що сегмент знаходиться в пам'яті і зміщення потрапило в потрібний інтервал, тоді система Pentium додає 32-розрядне поле База (Base) в дескрипторі до зміщення, формуючи те, що називається **лінійною адресою**, як показано на рис. 10.29.

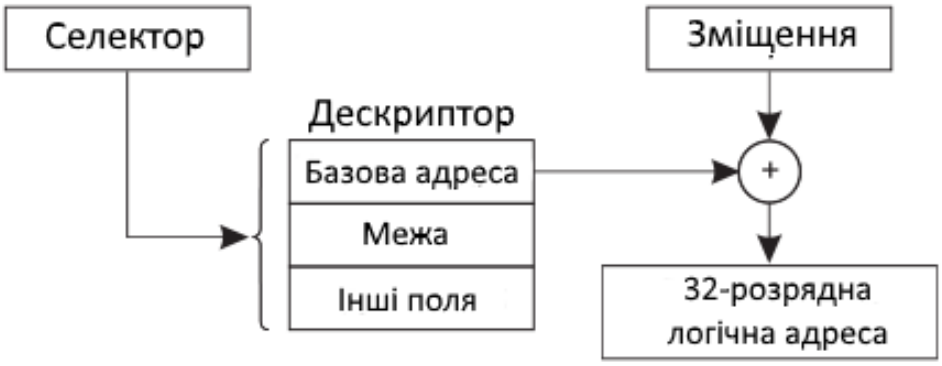


Рисунок 10.29 – Перетворення пари селектор-зміщення в лінійний адрес

Поле **База** розбито на три частини, які розкидані по дескриптору для сумісності з процесором Intel 80286, в якому поле **База** має тільки 24 біта. По суті, поле **База** дозволяє кожному сегменту починатися в довільному місці всередині 32-розрядного лінійного адресного простору.

Якщо розбиття на сторінки відключене, то лінійна адреса інтерпретується як фізичнна адреса і відправляється в пам'ять для читання або запису. Якщо розбиття на сторінки використовується, то лінійна адреса розглядається як віртуальна і відображається у фізичну адресу з використанням таблиць сторінок. Таким чином, при відключеній сторінковій схемі пам'яті отримуємо чисту схему сегментації.

Також слід зазначити, що ця модель працює і в тому випадку, коли деякі додатки не вимагають сегментації, а просто задовольняються єдиним, розбитим на сторінки 32-розрядним адресним простором. Усі сегментні регістри можуть бути налагоджені тим же самим селектором, у дескрипторі якого поле Base = 0, а поле Limit встановлене на максимум. Тоді зміщення команди буде лінійною адресою і використовуватиметься тільки один адресний простір, що приведе до звичайної сторінкової організації пам'яті. Фактично таким чином працюють усі сучасні операційні системи для комп'ютерів x86.

З іншого боку, якщо включено підкачування сторінок, лінійна адреса інтерпретується як віртуальна і відображається на фізичну адресу за допомогою таблиці сторінок практично так само, як в попередніх прикладах. Єдине реальне утруднення полягає в тому, що при 32-розрядній віртуальній адресі і сторінці розміром 4 Кб сегмент може містити 1 млн сторінок (1 Мб = $2^{20}$ = 4Гб/4Кб), тому використовується дворівневе відображення з метою зменшення розміру таблиці сторінок для невеликих сегментів.

Віртуальна адреса як і раніше є парою: селектор, який визначає номер віртуального сегменту, і зміщення всередині цього сегменту. Перетворення віртуальної адреси виконується в два етапи: спочатку працює сегментний механізм, а потім результат його роботи поступає на вхід сторінкового механізму, який і обчислює шукану фізичну адресу.

Робота сегментного механізму в даному випадку багато в чому повторює його роботу при відключеному сторінковому механізмі. На підставі значення індексу в селекторі вибирається потрібний дескриптор з таблиці GDT або LDT. З дескриптора витягається базова адреса сегменту і складається зі зміщенням. Дескриптори і таблиці мають ту ж структуру. Проте є і принципова відмінність, яка полягає в інтерпретації утримуваного поля базової адреси в дескрипторах сегментів. Якщо раніше дескриптор сегменту містив базову адресу сегменту у фізичній пам'яті і при складанні цієї адреси зі зміщенням з віртуальної адреси виходила фізична адреса, то тепер дескриптор містить базову адресу сегменту у віртуальному адресному просторі, і в результаті його складання зі зміщенням отримуємо лінійний віртуальний адрес.

Результуюча лінійна 32-розрядна віртуальна адреса передається сторінкового механізму для подальшого перетворення. Виходячи з того що розмір сторінки дорівнює 4 Кб (212), в адресі можна легко виділити номер віртуальної сторінки (старші 20 розрядів) і зміщення в сторінці (молодші 12 розрядів). Для відображення віртуальної сторінки у фізичну досить побудувати таблицю сторінок, кожен елемент якої містив би номер відповідної їй фізичної сторінки і її атрибути.

У процесорі Pentium так і зроблено, структура дескриптора сторінки показана на рис. 10.30. Крім номера сторінки (20 розрядів) дескриптор сторінки містить також поля, близькі за змістом відповідним полям дескриптора сегмента:

– P – біт присутності сторінки у фізичній пам'яті;
– W – біт дозволу запису в сторінку;
– U – біт – користувач/супервізор;
– A – ознака доступу, що мав місце, до сторінки;
– D – ознака модифікації вмісту сторінки;
– AVL – резерв для потреб операційної системи;
– PWT і PCD – біти, які управляють механізмом кешування сторінок.

| 20 | 3 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|
| № сторінки | AVL | 0 | D | A | PCD | PWT | U | W | P |

Рисунок 10.30 – Формат дескриптора сторінки

Таблиця сторінок може займати в пам'яті дуже значне місце – 4 байт*1 Мб = 4 Мб. Тому таблицю сторінок в силу її великого об'єму конструюють з двох рівнів: каталог таблиць сторінок (розділи) по 1024 дескриптори і таблиці сторінок, що містить 1024 32-разрядних записів [28]. Один розділ займає одну сторінку (1024*4 байт – 4 Кб), тобто, 1024 розділи. Записи в каталозі вказують на таблицю сторінок, а записи в таблицях сторінок вказують на сторінкові блоки (рис. 10.31).

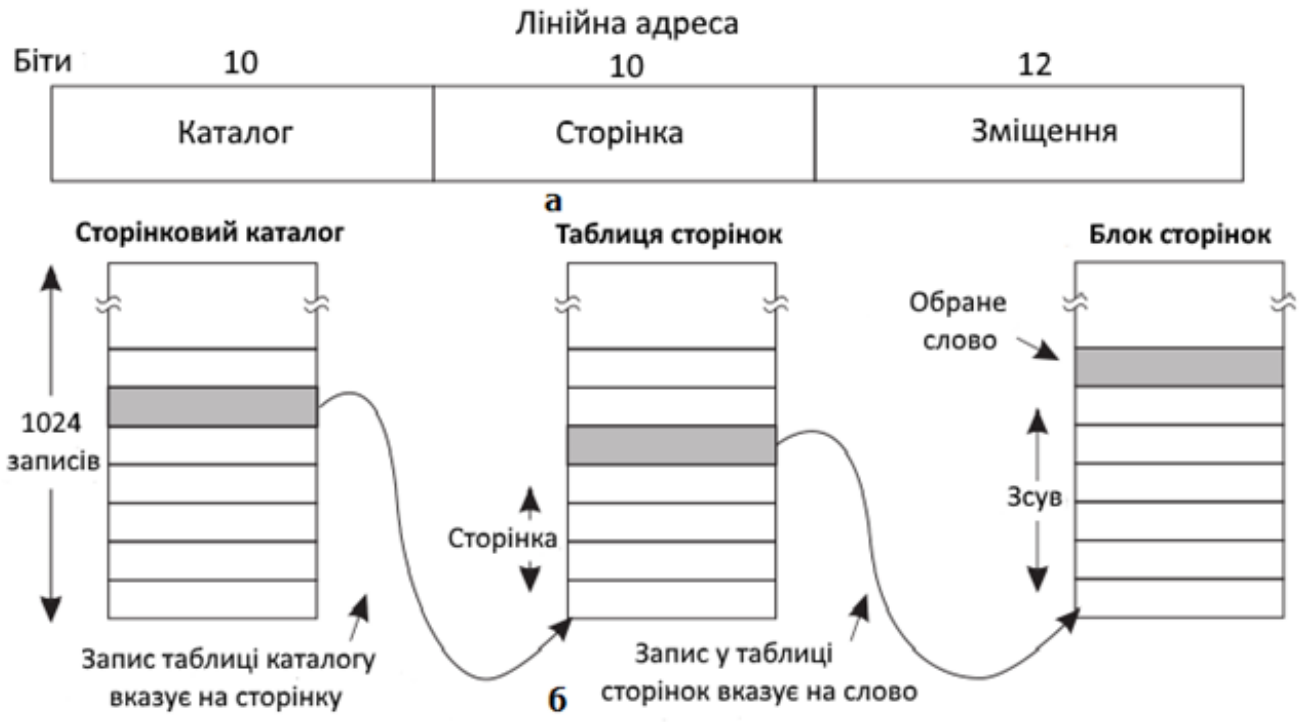


Рисунок 10.31 – Схема відображення лінійної адреси на фізичну

Лінійна адреса ділиться на три поля: **Каталог**, **Сторінка** і **Зміщення**. Поле **Каталог** використовується як індекс у сторінковому каталозі, що визначає розташування покажчика на правильну таблицю сторінок. Поле **Сторінка** використовується як індекс в таблиці сторінок, щоб знайти фізичну адресу сторінкового блоку. Щоб отримати фізичну адресу, до адреси сторінкового блоку додається останнє поле **Зміщення**.

Поле номера віртуальної сторінки (старші 20 розрядів) ділиться на дві рівні частини по 10 розрядів – поле номера розділу і поле номера сторінки в розділі. На підставі заданого в регістрі CR3 номера фізичної сторінки, що зберігає таблицю розділів, і зміщення в цій сторінці, що задається полем номера розділу, процесор знаходить дескриптор віртуальної сторінки розділу.

Відповідно до атрибутів цього дескриптора визначаються права доступу до сторінки, а також наявність її у фізичній пам'яті. Якщо сторінки немає в оперативній пам'яті, то відбувається переривання, в результаті якого ОС повинна виконати завантаження необхідної сторінки в пам'ять.

Віртуальні сторінки таблиці сторінок, як і усі інші сторінки, можуть вивантажуватися на диск. Віртуальна сторінка, що зберігає таблицю розділів (сторінковий каталог), завжди знаходиться у фізичній пам'яті.

Кожен запис в таблиці сторінок має розмір 32 біти, 20 з яких містять номер сторінкового блоку. Інші біти включають біти доступу і біт зміни сторінки, що встановлюються апаратурою для операційної системи, біти захисту і інші службові біти (рис. 10.32) [28].

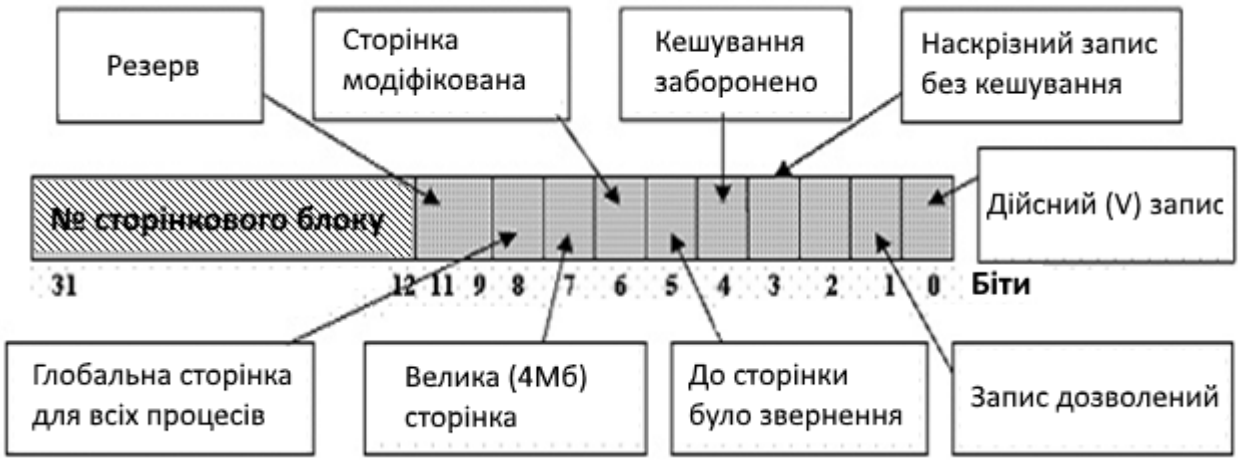


Рисунок 10.32 Структура рядка таблиці сторінок

Щоб уникнути повторних звернень до пам'яті, система x86, як і система MULTICS, має невеликий буфер швидкого перетворення адреси (TLB), який безпосередньо відображає найчастіші комбінації Каталог – Сторінка на фізичну адресу сторінкового блоку. Механізм, показаний на рис. 10.31, задіюється лише за відсутності поточної комбінації в буфері TLB, при цьому сам буфер оновлюється. Якщо відсутність потрібної інформації в буфері TLB зустрічається досить не часто, то система досягає непоганої продуктивності [9].

Перетворення лінійної віртуальної адреси у фізичну відбувається таким чином (рис. 10.33).

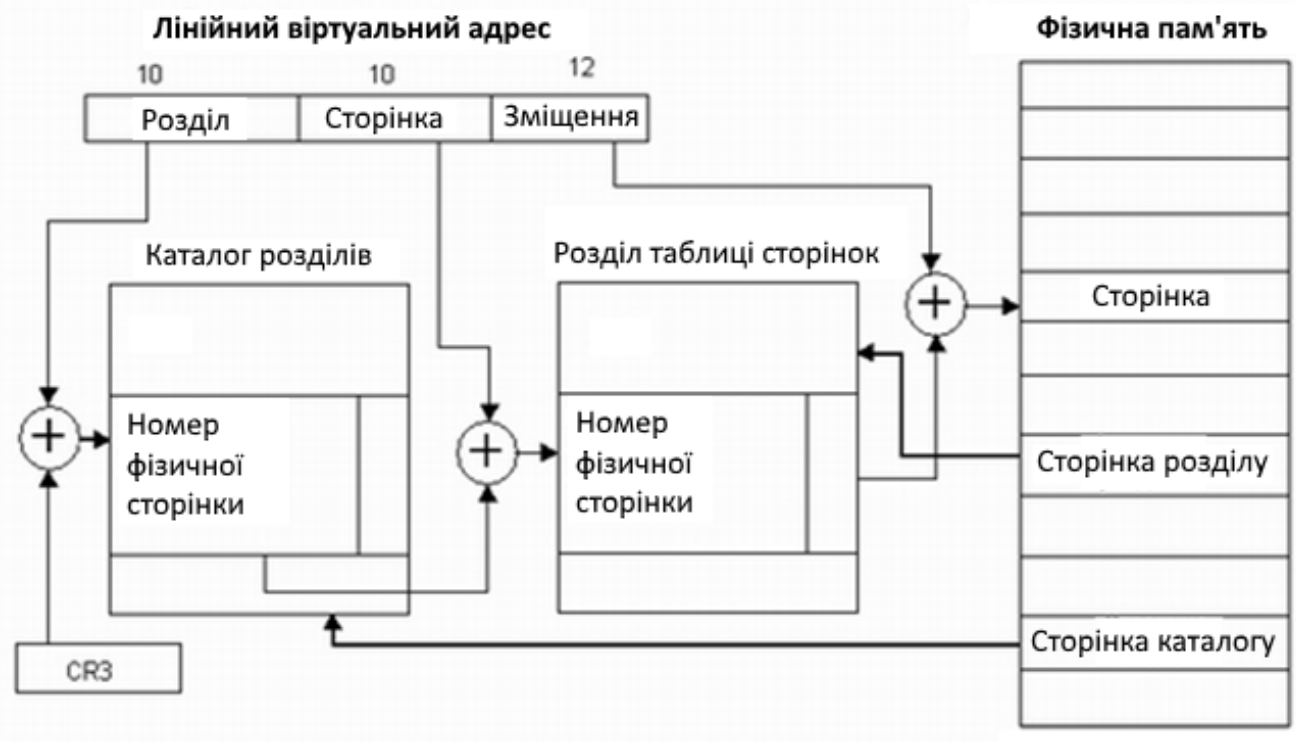


Рисунок 10.33 – Перетворення віртуального адреса у фізичну адресу

Сегментно-сторінкова організація пам'яті у Windows показана на рис. 10.34.

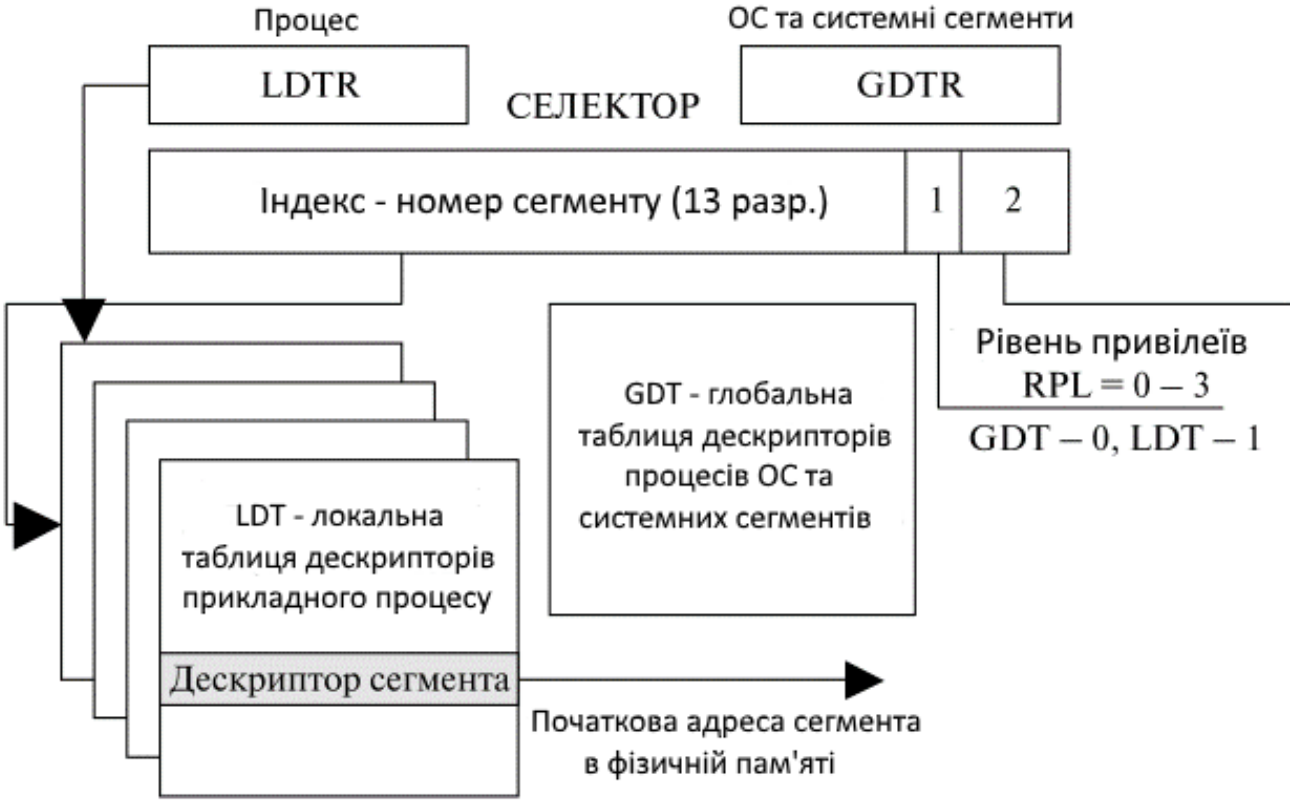


Рисунок 10.34 – Сегментно-сторінкова організація пам'яті у Windows

### 10.5.3 Свопінг

Різновидом віртуальної пам'яті є **свопінг (swapping)** або звичайне підкачування. Необхідною умовою для виконання задачі є завантаження її в оперативну пам'ять, об'єм якої обмежений. Для підвищення завантаження процесора був запропонований метод організації обчислювального процесу, що називається свопінгом (рис. 10.35). Процеси, що знаходяться в стані очікування, тимчасово цілком вивантажуються на диск. Планувальник ОС не виключає їх зі свого розгляду, і при настанні умов активізації деякого процесу цей процес переміщається в оперативну пам'ять. Якщо вільного місця в оперативній пам'яті бракує, то вивантажується інший процес. У разі переривання роботи процесу він переміщається назад у зовнішню пам'ять.

Якщо на комп'ютері одночасно виконується велике число обчислювальних задач і задач з інтенсивним введенням/виведенням, то вдається добитися високої ефективності використання центрального процесора.

Свопінг є окремим випадком віртуальної пам'яті і, отже, простіший в реалізації спосіб спільного використання оперативної пам'яті і диска. Проте підкачуванню

властива надмірність. Коли ОС вирішує активізувати процес, для його виконання, як правило, не вимагається завантажувати в оперативну пам'ять усі його сегменти повністю. Крім того, системи, що підтримують свопінг, мають ще один дуже суттєвий недолік: вони не здатні завантажити для виконання процес, віртуальний адресний простір якого перевищує наявну вільну пам'ять.

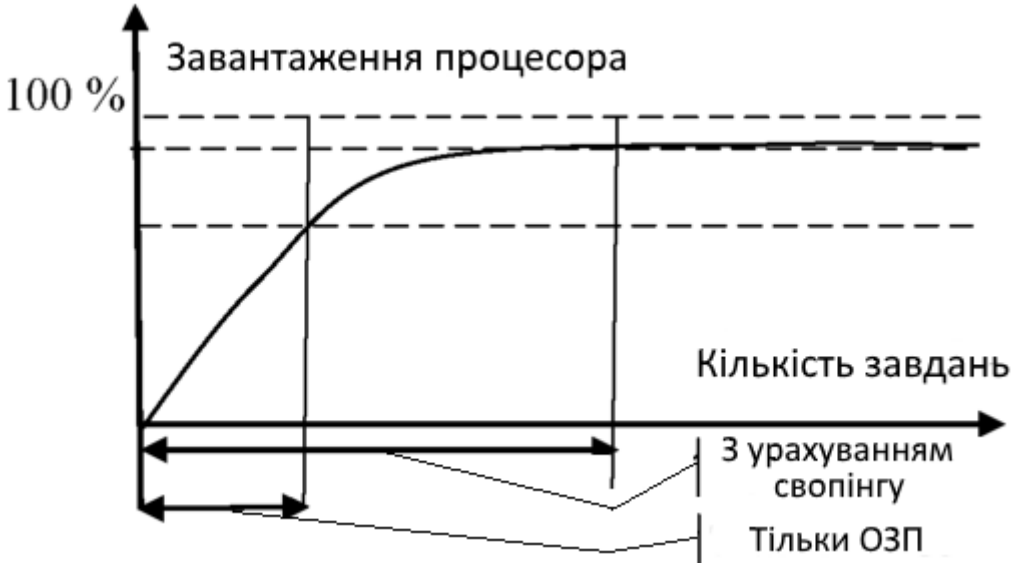


Рисунок 10.35 – Завантаження процесора при використанні свопінгу

Враховуючи ці недоліки «чистого» свопінгу, особливо у великих втратах часу на завантаження або вивантаження процесів, в сучасних операційних системах використовуються модифіковані варіанти свопінгу. Так, наприклад, у багатьох версіях операційної системи UNIX свопінг включається тільки в тому випадку, коли кількість процесів у пам'яті стає занадто великою.