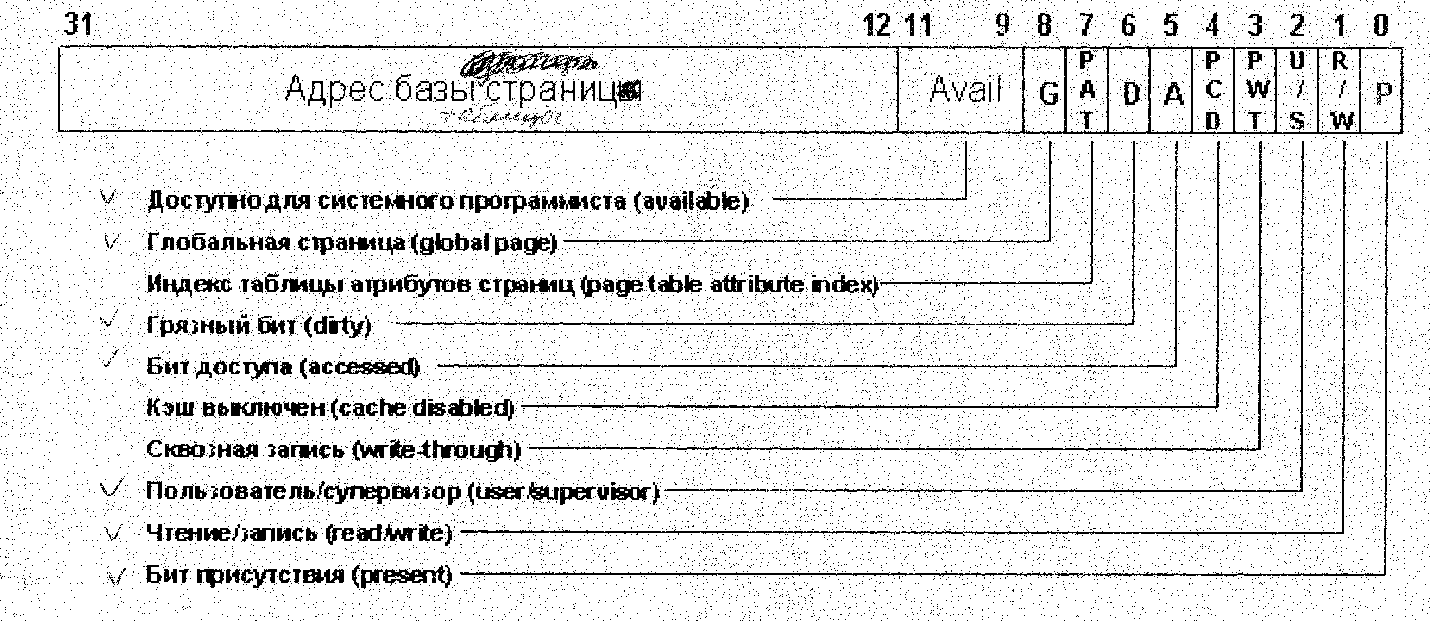
**Структура елементів PDE, PTE.**

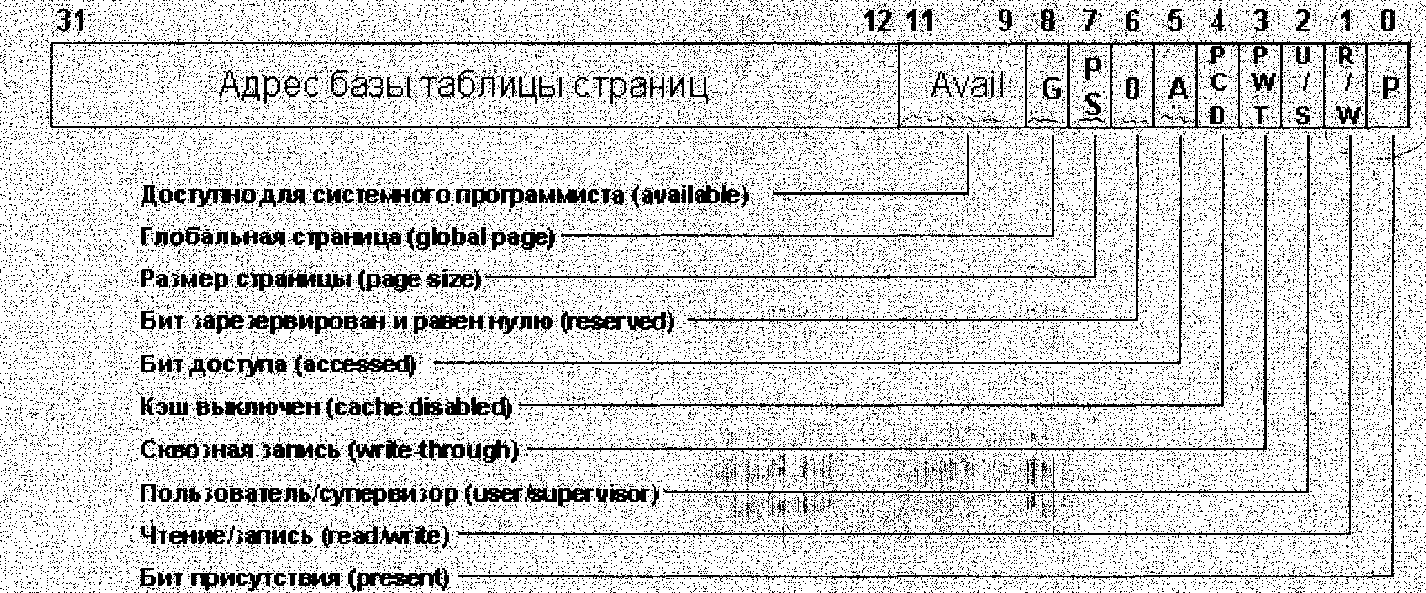
Таким чином, маємо, що при сторінковій організації пам’яті лінійна адреса (база з дескриптора + offset) не відповідають фізичній (тобто адресі, яку процесор виставляє на адресну шину). Лінійна адреса ділиться на три частини у випадку 4Кб сторінок, і на дві - у випадку 4Мб страниц.

4КБ сторінки. Структура Page Directory Entry (розряди 22-31 у лінійній адр.)

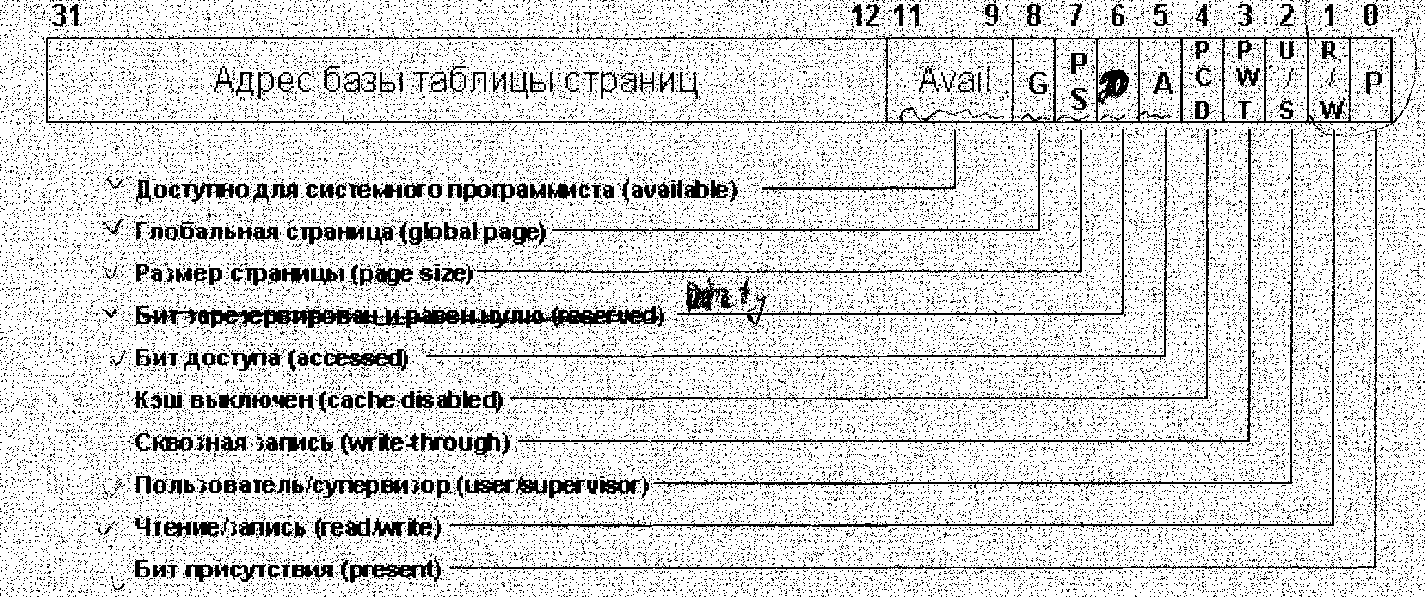
****

Адрес страницы таблицы страниц

4КБ сторінки. Структура Page Table Entry (розряди 12-21 у лінійній адр.)

****

4MБ сторінки. Структура Page Table Entry (розряди 22-31 у лінійній адр.)



Reserved=0

0

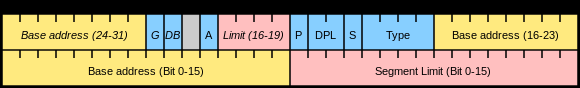
Р-флаг, біт 0: вказує, чи знаходиться сторінка (або таблиця сторінок) у фізичній пам'яті. Якщо він дорівнює 1 - сторінка в пам'яті, можна проводити формування фізичної адреси, якщо ж він скинутий - процесор викликає виключення #PF (Page Fault), обробник якого повинен завантажити її в пам'ять. Загальна інформація: процесор ніколи не міняє цей флаг сам - цим повинна займатися ОС.

R/W-флаг, біт 1: визначає привілеї читання/запису для сторінки або групи сторінок (у випадку, коли елемент каталога сторінок указує на таблицю сторінок). Якщо флаг скинуто - сторінка доступна ТІЛЬКИ ДЛЯ ЧИТАННЯ. Якщо встановлено - для читання і запису. Даний флаг пов'язаний з флагами U/S і WP з регістра CR0.

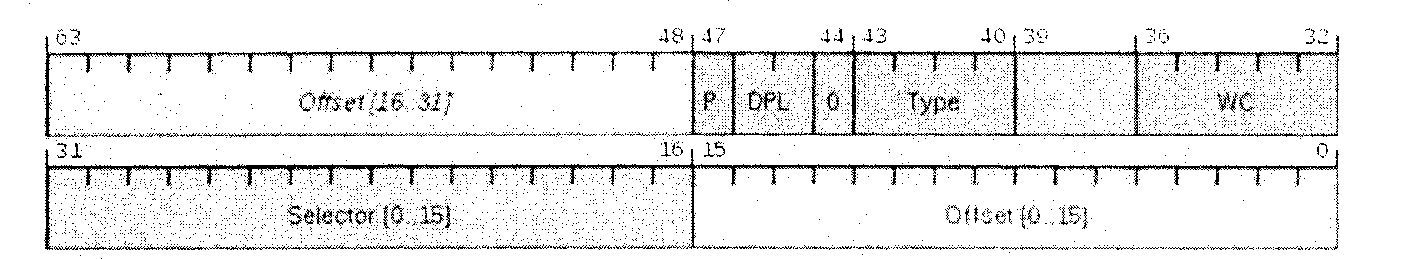
U/S-флаг, біт 2: визначає привілеї користувача/супервізора для сторінки або групи сторінок. Якщо флаг скинуто - сторінка доступна тільки для рівня привілеїв супервізора, якщо встановлено - для користувача та супервізора. Даний флаг пов'язаний з флагами U/S і WP.

**Структура дескриптора шлюзів. Таблиця ІDT.**

**Мал. 1.**  Структура звичайного дескриптора фрагмента ОП



**Дескриптор шлюзу** — службова структура, яка використовується для різних переходів.



**Мал. 2.**  Структура дескриптора шлюзів

**Зсув** (ангп. **Offset**, жовті поля) — зсув адреси точки входу в процедури (у сегменті коду);

**Селектор** (англ. **Selector**, помаранчеве поле) — селектор сегменту коду процедури, на яку відбувається перехід. Може бути сегментом коду або TSS; **Кількість параметрів** (англ. **Words count**, блакитне поле, біти 32-36) — кількість слів (16-розрядний стек) або подвійних слів (32-розрядний стек), що копіюються із стека головної програми в стек процедури, що викликається. Актуально тільки при зміні рівня привілеїв; використовується тільки в шлюзах виклику (Call Gate)

**Тип/права доступу** (блакитні поля, біты 40-47) — права доступу до шлюзу і його тип.

**Шлюз виклику** (англ. **CallGate**)

• Не можна використовувати в IDT.

• Характерною особливістю шлюзу виклику є наявність поля WC, завдяки якому можлива передача до 25 = 32 слів/подвійних слів у процедуру, що викликається, за допомогою стеку.

**Шлюз задачі** (англ. **TaskGate**)

• Може використовуватися в будь-якій з трьох дескрипторних таблиць.

• Єдина особливість цього шлюзу полягає в тому, що у якості фрагменту вказується фрагмент для збереження TSS (дескриптор TSS). Поля зсуву не використовуються і можуть мати будь-яке значення.

**Шлюз переривання** (англ. **InterruptGate**)

• Цей шлюз використовується тільки в IDT.

• Його особливістю є скидання флага переривань при вході в процедуру обробки. Оскільки попереднє значення регістра флагів зберігається в стеку, то при виході з процедури обробки флаг IF приймає початкове значення. Це дозволяє обробляти різні переривання.

**Шлюз пастки** (англ. **TrapGate**)

• Цей шлюз використовується тільки в IDT.

• Найпростіший шлюз. Під час обробки переривання використовується для переходу на процедуру обробки (за необхідності проводиться перемикання стеку).

**Таблиця ІDT: шлюз задачі, шлюз переривання, шлюз пастки.**

Використовується для забезпечення обробки переривань (апаратних і програмних) та виключень (логічних переривань) у процедурах-обробниках.

*Переривання* – запити на виконання деяких дій для обслуговування. Переривання бувають апаратні (або зовнішні) і програмні.

*Зовнішні переривання* генеруються периферійними пристроями, коли для них потрібно забезпечити запити на обслуговування.

Переривання, які генеруються під час виконання програм і які визначаються як запити на обслуговування однієї програми за допомогою іншої, називають *програмними перериваннями*.

*Виключення* є для процесора внутрішніми подіями і сигналізують про які-небудь помилкові умови при виконанні тієї або іншої інструкції.

*Виключення* процесора, залежно від способу генерації і можливості рестарту команди, що викликала виключення, діляться на порушення, пастки і аварії.

***Порушення*** (відмова) - це *виключення*, яке виявляється або перед виконанням, або під час виконання команди. При цьому процесор переходить в стан, що дозволяє здійснити рестарт команди. У якості адреси повернення у стек обробника заноситься адреса команди, що викликала виключення.

***Пастка*** виникає на межі команд відразу ж після команди, що викликала це *виключення*. Значення регістрів CS і EIP, що заносяться в стек обробника, указують на наступну команду. Наприклад, якщо пастка спрацювала на команді JMP, то в стеку запам'ятовуються значення регістрів CS і EIP, що вказують на посилання команди JMP.

***Аварія*** не дозволяє здійснити рестарт програми (серйозна помилка), часто неможливо локалізувати команду, яка викликала конкретне *виключення*.

Де розташовується таблиця переривань IDT? Місце її розташування визначається вмістом 6-байтового (48 біт) внутрішнього регістра процесора IDTR. Формат регістра IDTR повністю аналогічний формату регістра GDTR, для його завантаження використовується команда LIDT, для запису - SIDT. Так само, як регістр GDTR містить 32-бітову фізичну адресу таблиці GDT і її 16-бітний ліміт (границю), так і регістр IDTR містить 32-бітову фізичну адресу дескрипторної таблиці переривань IDT та її межі (16 біт).

Регістр IDTR зазвичай завантажують перед переходом в захищений режим. Зрозуміло, що відповідна задача має працювати в привілейованому нульовому кільці.

Кожна сторінка описується елементами РТЕ (12 молодших бітів). Кожна сторінка характеризується трьома атрибутами Т (transition), D (dirty), P (present). Процеси (задачі) ОС Windows мають окремі каталоги сторінок, які утворюють *робочу множину процесу*. В Windows використовується утиліта ядра VMM (Virtual Memory Manager), який дозволяє організувати управління сторінками довільної задачі. VMM утворює та зберігає базу даних усіх сторінок пам’яті, які є у розпорядженні ОС (мал.3).

31 27 26 7 6 3 2 0

|  |  |  |  |
| --- | --- | --- | --- |
| Захист | Фізична адреса | Файл підкачки | Стан |

Мал.3. Структура елементу бази даних сторінок VMM.

Біти Т, D, P дозволяють організувати *swapping* сторінок, а також цікавий процес крадіжки сторінок з робочої множини. Якщо сторінка є активною сторінкою робочої множини процесу (тобто використовується), Т=0, D=0 або 1, в залежності від проведений дій, Р=1 (VALID page). Перевіркою, що проводиться час від часу, визначається група сторінок, які не використовуються деякий час. Такі сторінки позначаються Т=1 (перехідні), Р=1 (присутні), D=0 (StandBy) або D=1 (Notified). Якщо за визначений час не відбувається звернення до таких сторінок, вони переписуються у swap-файл (файл підкачки), а біт Р=0. Ознака Р=0 для сторінки в робочій множині означає, що її зміст у swap-файлі, а сама вона може бути перерозподілена іншій задачі (FREE). При потребі (звертанні до цієї сторінки, ситуація #GP), відбувається підкачка змісту сторінки зі swap-файлу.

**Механізми захисту**

У захищеному режимі процесор забезпечує певні механізми захисту на основі сегментації і на основі сторінкового перетворення. Механізми захисту дозволяють обмежувати доступ до певних сегментів або сторінок за допомогою рівнів привілеїв (4 для фрагментів і 2 для сторінок). Наприклад, критичні код і дані операційної системи можуть бути розташовані на більш привілейованому рівні, ніж прикладні програми. Це дозволить обмежити і контролювати доступ прикладних програм до функцій операційної системи.

Механізм захисту забезпечує відповідність будь-якого посилання на елементи пам'яті певним умовам. Всі перевірки виконуються до початку циклу звернення до пам'яті. Порушення будь-якої умови приводить до генерації виключення. Перевірки виконуються паралельно з формуванням адреси і тому не погіршують продуктивність процесора. Всі посилання повинні пройти наступні перевірки:

* контроль границі (межі фрагментів);
* контроль типу;
* контроль рівня привілеїв;
* контроль вирівнювання;
* обмеження адресного простору;
* обмеження точок входу в процедури (для шлюзів);
* обмеження набору команд (привілейовані інструкції).

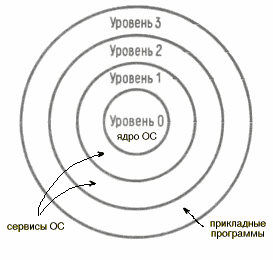
У захищеному режимі немає способу відключити механізм захисту. Навіть якщо призначити всім фрагментам і задачам нульовий (самий вищий) рівень привілеїв, все одно виконуватимуться перевірки контролю межі та типу. В той же час, механізм захисту на рівні сторінок можна подавити, якщо призначити всім сторінкам рівень привілеїв суперкористувача і забезпечити доступ на читання і запис.

Контроль меж і типів фрагментів забезпечує цілісність сегментів коду і даних. Програма не має права звернутися до віртуальної пам'яті, що виходить за межу того або іншого фрагменту. Програма не має права звернутися до сегменту даних як до коду, і навпаки.

Архітектура захисту МП забезпечує 4 ієрархічних рівня привілеїв, що дозволяє обмежити задачі доступ до окремих фрагментів в залежності від її поточних привілеїв. Привілеї - це властивість (встановлюється на етапі проектуванні системи), яка визначає, які комп'ютерні операції вирішуються у будь-який момент часу і які способи доступу до пам'яті є законними. Привілеї використовуються для забезпечення безпеки в комп'ютерній системі. Привілеї реалізуються шляхом присвоювання значень від 0 до 3 об'єктам, які розпізнаються процесором. Значення 0 відповідає найбільшим привілеям, тоді як значення 3 - найменшим.

Чотири рівні привілеїв можна інтерпретувати у вигляді кілець захисту (мал. 3). Центр (рівень 0) призначений для фрагментів, що містять найбільш критичні програми (ядро операційної системи). Зовнішні кільця призначені для фрагментів з менш критичними програмами або даними. Використання всіх чотирьох рівнів привілеїв не є необхідним. Існуючі системи, спроектовані з меншою кількістю рівнів, можуть просто ігнорувати інші допустимі рівні. UNIX і Windows, наприклад, використовують тільки два рівні привілеїв: 0 (для ядра системи) і 3 (для всього іншого).

**Мал. 4.**  Інтерпретація рівнів привілеїв у вигляді кілець захисту



Механізм контролю рівня привілеїв мікропроцесора оперує значеннями:

**CPL** - поточний рівень привілеїв (Current Privilege Level): рівень привілеїв, на якому в даний момент виконується задача. Значення CPL зберігається в полі RPL селектора сегменту коду, який розміщений в регістрі CS. Це значення відповідає рівню привілеїв дескриптора фрагменту коду, що виконується. Рівень привілеїв змінюється, коли управління передається сегменту коду з іншим значенням DPL (за винятком підлеглих сегментів коду).

**DPL** - рівень привілеїв дескриптора (Descriptor Privilege Level): найменш привілейований рівень, на якому задача може отримати доступ до фрагменту або шлюзу, пов'язаного з цим дескриптором. Рівень DPL визначається бітами 46 і 45 дескриптора.

**RPL** - запрошуваний рівень привілеїв (Requested Privilege Level), що використовується для тимчасового пониження свого рівня привілеїв при зверненні до пам'яті. RPL заноситься в молодші біти селектора.

Висновок 1.

*Таким чином, підсумуємо. Будь-який селектор має поле RPL (биті 0 і 1). У регістрі CS зберігається селектор поточного виконуваного сегменту коду. Цьому селектору відповідає дескриптор в таблиці GDT або LDT. У дескрипторі, в блоці доступу дескриптора розташовується поле DPL (биті 45 і 46). Вказані поля беруть участь в механізмі захисту пам'яті.*

*Коли операційна система готує програму для запуску, вона формує в GDT або LDT дескриптор, що описує сегмент коду програми. У цьому дескрипторі в полі DPL байта доступу проставляється номер кільця, в якому працюватиме дана програма - поточний рівень привілеїв CPL (Current Privilege Level). Тобто можливості програми визначаються вмістом поля DPL в байті доступу.*

*Поточний рівень привілеїв копіюється в полі RPL селектора сегменту коду, завантаженого в регістрі CS. Це дозволяє задачі проаналізувати поточний рівень привілеїв, виходячи із значення поле RPL в регістрі CS. Проте задача не може змінити свій рівень привілеїв простій заміною вмісту поля RPL в сегменті CS.*

Механізм контролю рівня привілеїв зазвичай порівнює рівень привілеїв дескриптора (DPL) з максимальним з двох чисел CPL і RPL. Найбільш привілейований з поточного рівня привілеїв і запрошуваного вважається ефективним рівнем привілеїв: EPL = max (CPL, RPL).

Контроль привілеїв при доступі здійснюється при завантаженні селектора в сегментний регістр DS, ES, FS, GS (або при зверненні до пам'яті, якщо селектор міститься в коді інструкції). Програма може звернутися до фрагменту даних, який знаходиться на тому ж або нижчому рівні привілеїв (з урахуванням RPL), тобто доступ до даних дозволений, якщо max(CPL, RPL) <= DPL; інакше генерується порушення загального захисту.

Контроль привілеїв при доступі до стека здійснюється при заванта-женні селектора в регістр SS. Програма повинна використовувати сегмент стека, що знаходиться на тому ж рівні привілеїв, тобто CPL = RPL = DPL.

Для передачі управління на звичайний сегмент коду його рівень привілеїв повинен співпадати з поточним рівнем привілеїв. Значення RPL повинне бути не більше CPL, щоб не викликати виключення, але незалежно від значення RPL рівень привілеїв не зміниться: CPL = DPL. Поточний рівень привілеїв може змінитися при передачі управління через селектор шлюзу (останні два типи не використовуються в командах переходу і виклику). При передачі управління через селектор шлюзу задачі або селектор TSS відбувається перемикання задач. Одна задача може передати управління іншій задачі, при цьому контролюється, щоб об'єкт, через який передається управління, був на тому ж або нижчому рівні привілеїв, ніж CPL (правило контролю як для сегментів даних).

Висновок 2.

*Дескриптори, що описують сегменти даних, містять поле рівня привілеїв дескриптора DPL (Descriptor Privilege Level). Поле DPL містить мінімальні привілеї, які потрібні для доступу до сегменту даних.*

*Перед тим, як звернутися до сегменту даних, програма повинна завантажити в один з сегментних регістрів селектор, відповідний потрібному сегменту даних. У селекторі необхідно вказати поле рівня запрошуваних привілеїв RPL (Requested Privilege Level).*

*Програмі буде наданий доступ до сегменту тільки у тому випадку, коли рівень привілеїв дескриптора запрошуваного сегменту DPL більше або рівний значенню max(CPL,RPL), тобто найбільшому із значень поточного рівня привілеїв CPL і рівня запрошуваних привілеїв RPL:*

*DPL >= max(CPL,RPL)*

*Якщо задача спробує отримати доступ до більш привілейованого, ніж вона сама, сегменту пам'яті, її виконання буде перервано.*

Крім того, поточний рівень привілеїв задачі впливає на можливість виконання тих або інших специфічних команд (привілейованих інструкцій) та інструкцій, результат виконання яких залежить від поля IOPL в регістрі флагів (I/O Privilege Level - рівень привілеїв вводу-виводу, 12-13 біти). Механізм захисту процесора дозволяє виконувати ці інструкції, тільки якщо задача володіє достатніми привілеями, тобто CPL <= IOPL.

Функції сторінкової трансляції забезпечують додаткові механізми захисту на рівні сторінок. Особливо це зручно при використанні суцільної моделі пам'яті, коли і операційна система, і прикладні програми працюють в єдиному просторі логічних адрес, оскільки захист на рівні сторінок може забезпечити розмежування привілеїв для сторінок операційної системи і прикладних програм. Захист на рівні сторінок надає два типи контролю: обмеження адресного простору (сторінки супервізора і сторінки користувача) і обмеження типу доступу (доступ тільки для читання і доступ на читання-запис). Якщо перевірки не проходять, генерується сторінкове порушення.

Обмеження адресного простору на рівні сторінок забезпечується двома рівнями привілеїв: режим супервізора (рівень 0) і режим користувача (рівень 1). Режим супервізора відповідає привілеям на рівні фрагментів CPL = 0, CPL = 1 і CPL = 2. У режимі супервізора доступні всі сторінки. Режим користувача відповідає привілеям на рівні сегментів CPL = 3. У режимі користувача доступні тільки сторінки користувача. У адресний простір супервізора потрапляють сторінки, для яких відповідний елемент таблиці сторінок або каталога таблиць сторінок містить біт U/S = 0. У адресний простір користувача потрапляють сторінки, для яких і відповідний елемент таблиці сторінок, і відповідний елемент каталога таблиць містять біт U/S = 1.

У МП Intel обмеження типу доступу застосовується тільки в адресному просторі користувача. Програма в режимі користувача має право змінювати тільки ті сторінки, для яких і відповідний елемент таблиці сторінок, і відповідний елемент каталога таблиць містять біт R/W = 1. Якщо для якої-небудь сторінки елемент таблиці сторінок або елемент каталога таблиць містить біт R/W = 0, то сторінка доступна лише для читання. У режимі супервізора всі сторінки доступні і для читання, і для запису.

Коли використовуються обидва види захисту і на рівні сторінок, і на рівні сегментів, процесор спочатку виконує перевірку захисту сегментів, і лише у разі успіху - перевірку захисту сторінок. Тобто, якщо доступ до пам'яті відкидається захистом на рівні сегментів, то це згенерує порушення загального захисту, а перевірки захисту сторінок виконуватися не будуть і додаткове сторінкове порушення не виникне. Якщо перевірки захисту сегментів проходять успішно, але порушуються правила захисту сторінок, генерується сторінкове порушення. Доступ до пам'яті надається, тільки коли виконані всі правила і для сегментів, і для сторінок.