34. 4 . 2. Охарактеризувати засоби підтримки віртуальної пам'яті на рівні сегментів у захищеному режимі.

Захищений режим ґрунтується на сегментній організації пам'яті. Сегменти в захищеному режимі, на відміну від реального режиму, можуть мати розмір в межах від 1 байта до 4 Гбайт. При 32-розрядній адресації використовуються всі 32 розряди зміщення в сегменті, тобто значення від 0 до 4 Гбайт-1 (2^{32}-1). Якщо для кожного сегменту пам'яті зберігати допустиме зміщення і порівнювати його зі зміщенням, сформованим в команді при звернені до сегменту, то при перевищенні зміщення над допустимим виникає можливість реалізації захисту від взаємного впливу між програмами через помилки в адресації. Це не застрахує від помилок усередині програми (сегменту), але не дає можливості помилкам "вирватись" за межі програми (сегменту). Де ж зберігати максимально допустиме зміщення в сегменті? Інженери ф. Intel вирішили зберігати його в спеціальній системній 8-байтній структурі даних в пам'яті, яка має назву **дескриптор сегменту**.

Загальна структура дескриптора сегменту

Байт 7						Байт 6								Байт 5								Байт 4							
Base_3 (31-24)						**G**	**D/B**	0	**U**	Limit_2 (19 - 16)				**P**	**DPL**			**S**	**I**	**C/E**	**R/W**	**A**	Base_2 (23-16)						
Base_1 (15 - 0)														Limit_1 (15 - 0)															
Байт 3						Байт 2								Байт 1								Байт 0							

В дескрипторі сегменту для максимального зміщення відводиться поле Limit, яке розбито на два під поля -Limit_1 (0-ий та 1-ий байт дескриптора) - молодша частина та Limit_2 (молодші чотири розряди 6-го байта дескриптора) - старші чотири розряди. Фактично в дескрипторі розміщується не розмір а границя - максимально можливе значення зміщення в сегменті. Тобто, щоб одержати розмір сегменту необхідно до значення поля Limit додати 1, наприклад, 0-ве значення поля Limit визначає сегмент розміром в 1 байт. В 20 розрядах поля Limit можна задати сегменти розміром лише в 1 Мбайт. Для задання більших розмірів використовується біт **G** - старший біт 6-го байта дескриптора. Якщо **G** =1, то максимальне зміщення в полі Limit задається не в байтах, а в 4 Кбайтних областях. Якщо через max_offset позначити максимально допустиме зміщення в сегменті, то при **G** =1 маємо:

max_offset = Limit*4096 + 4095

Оскільки розмір сегменту дорівнює max_offset+1, то при **G** =1 мінімальний розмір сегменту (Limit=0) дорівнює 4Кбайт, а максимальний (Limit=0fffffh):

$((2^{20}-1)* 2^{12} + 2^{12}-1) +1 = 2^{32}$=4Гбайт.

Наступною фундаментальною характеристикою сегменту є фізична адреса сегменту. В реальному режимі старші 16 біт фізичної адреси сегменту розміщались в сегментних регістрах. При наявності дескриптора природно розмістить в ньому і адресу сегменту. Для цього в дескрипторі відведено 32-розрядне поле Base, яке складається з під полів Base_1, Base_2 та Base_3 (мал.1). Таким чином дескриптор сегменту вказує як на розміщення сегменту в ОЗП, так і на його розмір.

Сукупність дескрипторів сегментів формується в таблицю, яка має назву "**Глобальна таблиця дескрипторів**". Для задання глобальної таблиці дескрипторів в процесорі служить спеціальний 48-розрядний регістр **GDTR**, старші 32 розряди якого вказують на фізичну адресу таблиці, а молодші 16 - на її розмір. Для завантаження регістра GDTR реалізована привілейована (системна) команда **Lgdt**. При наявності адреси таблиці, любий дескриптор в таблиці дескрипторів однозначно визначається його 16-розрядним зміщенням (адресою) в таблиці дескрипторів. В свою чергу кожний дескриптор визначає розміщення та розмір сегменту в ОЗП. В захищеному режимі в сегментні регістри записується зміщення

дескриптора сегменту в таблиці дескрипторів, яке разом із вмістом регістра GDTR повністю і однозначно визначає сегмент пам'яті. Оскільки всі дескриптори є 8-байтними, то зміщення любого дескриптора в таблиці дескрипторів кратне 8 і завжди має нульове значення в трьох молодших розрядах. Тому ці три розряди використовуються особливо - 2-й розряд (поле TI - поле таблиці) при одиничному значенні вказує, що адресується не глобальна, а локальна таблиця дескрипторів (див. рекомендовану літературу - в даній лабораторній роботі локальні таблиці дескрипторів не використовуються). А два молодші розряди - це поле RPL, яке визначає бажаний рівень привілеїв (в даній лабораторній роботі в основному RPL=0, бажаний рівень привілеїв є системним). Поєднання зміщення дескриптора з полями TI та RPL дістало назву **селектора сегменту**, який фактично і завантажується в сегментні регістри. Звідси зрозуміла фундаментальна різниця між програмами в реальному та захищеному режимах, яка полягає в трактовці вмісту сегментних регістрів. В реальному режимі сегменті регістри містять старші 16 розрядів фізичної адреси сегменту, а в захищеному режимі - селектор сегменту. Таким чином, люба програма реального режиму, яка завантажує вміст сегментних регістрів, в захищеному режимі є непрацездатною. Якщо ж якась процедура, розроблена для реального режиму, не завантажує сегментні регістри новими значеннями, то вона буде працездатною і в захищеному режимі. Для можливості виконання будь-яких, раніше розроблених програм, в захищеному режимі було реалізовано спеціальний під режим віртуального процесора 8086, як окремої задачі захищеного режиму.

В відповідності до викладеного, при виконанні любої команди, яка звертається до пам'яті, необхідно попередньо по значенню селектора із сегментного регістру та вмісту регістра GDTR сформувати фізичну адресу дескриптора, прочитати із пам'яті значення поля Base, скласти його із сформованим в команді зміщенням і по одержаній фізичний адресі виконувати читання чи записування даних. Тобто кожна команда звернення до пам'яті супроводжується читанням із пам'яті полів дескриптора сегменту. Звичайно, це суттєво (в декілька раз) зменшить продуктивність процесора. Для запобігання непродуктивним втратам часу, сегментні регістри розширили до 80 розрядів (64 розряди для дескриптора сегменту). Це розширення сегментних регістрів носить назву **тіньових регістрів**. При любому завантаженні селектора в сегментний регістр, автоматично завантажується в тіньовий регістр дескриптор, на який вказує селектор. В подальшому, при любій роботі із вибраним сегментом, всі поля дескриптора цього сегменту знаходяться "під рукою" в процесорі. Звичайно, команди завантаження сегментних регістрів суттєво ускладнюються і в декілька раз збільшується час їх виконання. Але в реальних програмах команди завантаження сегментних регістрів зустрічаються не часто, тому вплив часу їх виконання на загальну продуктивність процесора буде дуже незначний.

ЗАУВАЖЕННЯ. В тіньові регістри поля дескриптора можна записати лише командою завантаження сегментного регістру, тобто змінить значення окремих полів дескриптора в тіньовому регістрі не можливо. Поля дескриптора програма може змінити лише в пам'яті (в таблиці дескрипторів). Але зміна полів в пам'яті автоматично не змінює їх в тіньовому регістрі. Для того, щоб зміна полів дескриптора стала дієвою, необхідно пере завантажить сегментний регістр тим самим селектором, який в ньому був, наприклад:

```
Mov   ax,ds
Mov   ds,ax
```

що для непосвячених виглядає достатньо абсурдно.

Розміщення дескрипторів сегментів, з якими виконується робота, в тіньових регістрах "під рукою" процесора, дозволяє досить просто встановлювати і контролювати різноманітні правила захисту. Насамперед не викликає ускладнень контроль зміщення в сегменті на відповідність поля Limit. Крім того, 5-ий байт дескриптора, який названо як байт атрибутів (AR), дозволяє значно деталізувати властивості сегменту. Розглянемо окремі поля (розряди) цього байту детальніше.

Старший біт байта атрибутів (біт **P - біт присутності**) використовується як ознака присутності сегменту в ОЗП. Це дозволяє на рівні операційної системи організувати

"підкачку" сегментів з жорсткого диску, звільнюючи пам'ять від уже використаних сегментів шляхом їх запису, наприклад, на жорсткий диск. Щоб операційна система не "відправила" на жорсткий диск сегмент, який в той момент використовується, то при завантажені сегментного регістра селектором, апаратура процесора записує 1 в молодший біт байта атрибутів (біт **A - біт доступу**) відповідного дескриптора. Операційна система може використовувати для своїх потреб 4-ий біт 6-го байта дескриптора (біт **U** – **біт користувача**), який апаратурою процесора не встановлюється і не використовується.

Біти 6 та 5 байта атрибутів утворюють поле **DPL - рівень привілеїв** сегменту. Детальніше система привілеїв буде розглянута в наступному підрозділі. Біт 4 байта атрибутів (біт **S - системний**) в дескрипторах сегментів завжди встановлюється в 1. Значення **S** =0 використовується для ідентифікації спеціальних системних дескрипторів, частина із яких буде розглянута при організації переривань в захищеному режимі. Всі чотири молодші біти байта атрибутів в системному дескрипторі використовуються для вказівки типу дескриптора. В дескрипторах сегментів для вказівки на тип дескриптора використовуються біти 3-1 байта атрибутів. При цьому 3-й біт, (біт **I - біт призначення**) вказує на сегмент кодів (**I** =1), або на сегмент даних або стека (**I** =0).

Структура дескриптора сегмента кодів

Байт 7	Байт 6	Байт 5	Байт 4
Base_3 (31-24) **G** **D** 0 **U** Limit_2 (19 - 16)		**P** **DPL** 1 1 **C** **R** **A**	Base_2 (23-16)
Base_1 (15 - 0)		Limit_1 (15 - 0)	
Байт 3	Байт 2	Байт 1	Байт 0

В дескрипторі сегмента кодів 2-й біт байта атрибутів (біт **C – узгодженості**) визначає порядок між сегментної (дальньої) передачі управління на сегменти, які мають більший рівень привілеїв (при **C**=1 не змінюється поточний рівень привілеїв), а 1-ий біт (біт **R** – **читання**) дозволяє (при **R**=1), або забороняє (при **R**=0) командам читати із сегмента кодів. Крім того, в дескрипторі сегмента кодів 6-ий біт 6-го байта (біт **D** – **розмір**) визначає розрядність адрес та даних по умовчанню (**D**=0 – 16-розрядні, **D**=1 – 32-розрядні).

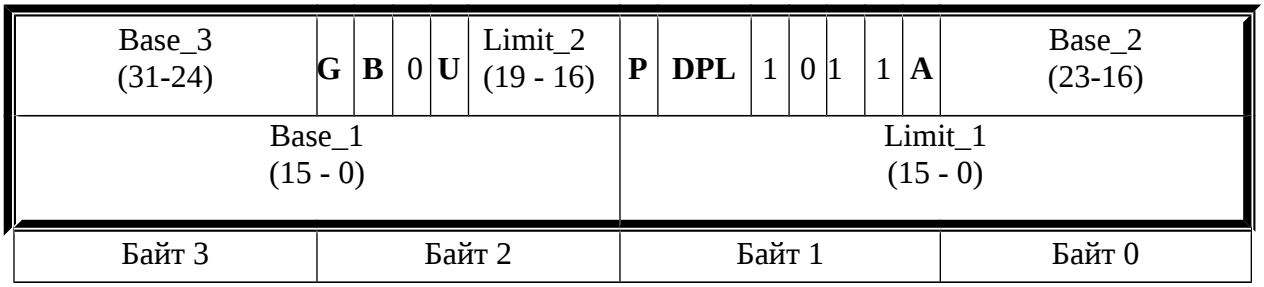
Структура дескриптора сегмента даних

Байт 7	Байт 6	Байт 5	Байт 4
Base_3 (31-24) **G** **B** 0 **U** Limit_2 (19 - 16)		**P** **DPL** 1 0 0 **W** **A**	Base_2 (23-16)
Base_1 (15 - 0)		Limit_1 (15 - 0)	
Байт 3	Байт 2	Байт 1	Байт 0

В дескрипторі сегмента даних 1-ий біт байта атрибутів (біт **W** – записування) дозволяє (**W** = 1), або забороняє (**W** = 0) записування в сегмент даних.

Структура дескриптора сегмента стека

Байт 7	Байт 6	Байт 5	Байт 4



Особливістю дескриптора сегменту стека є трактовка поля Limit. Поле ліміт в цьому дескрипторі визначає не максимально можливе значення зміщення в сегменті, а мінімально не можливе. Тобто дозволеними є зміщення від Limit+1 до maxsp. Значення maxsp визначається бітом **B** (6-им бітом 6-го байта дескриптора). Якщо **B**=0 , то maxsp=0FFFFH і в командах роботи із стеком використовується 16-розрядний регістр SP. Якщо **B**=1 , то maxsp=0FFFFFFFH і в командах роботи із стеком використовується 32-розряднийрегістр ESP.

Відповідно до значень розглянутих атрибутів апаратурою процесора контролюються наступні обмеження при виконанні команд:

a) завантаження в сегментні регістри DS, ES, GS, FS селектора дескриптора сегмента кодів, в якому заборонено читання даних;
b) завантаження в сегментний регістр SS селектора дескриптора сегмента з забороною запису даних;
c) завантаження в сегментний регістр CS селектора дескриптора сегмента, який не є кодовим;
d) любе записування в сегмент кодів;
e) читання із сегмента кодів, де читання заборонено;
f) записування в сегмент даних де заборонено запис:

Крім того, контролюється попадання зміщення в сегменті (ефективної адреси) в дозволений діапазон значень. Зауважимо, що в загальному випадку селектор сегмента даних може бути записаний в регістр SS, а селектор сегмента стека – в регістри DS, ES, GS, FS