ЗАХИЩЕНИЙ РЕЖИМ ВІРТУАЛЬНОЇ АДРЕСАЦІЇ 32-РОЗРЯДНИХ ПРОЦЕСОРІВ

1. Організація пам'яті 32-розрядних процесорів

Пам'ять для 32-розрядних процесорів 80x86 підрозділяється на байти (8 біт), слова (16 біт), подвійні слова (32 біт). Слова записуються в двох суміжних байтах, починаючи з молодшого. Адресою слова є адреса його молодшого байта. Подвійні слова записуються в чотирьох суміжних байтах, знов-таки починаючи з молодшого байта, адреса якого і є адресою подвійного слова.

Більш великими одиницями є сторінки і сегменти. Пам'ять може логічно організовуватися у виді одного або багатьох сегментів перемінної довжини (у реальному режимі - фіксованої). Сегменти можуть вивантажуватися на диск і в міру необхідності підкачуватися з нього у фізичну пам'ять. Крім сегментації, у захищеному режимі можлива розбивка (Paging) логічної пам'яті на сторінки розміром 4 Кбайт, кожна з яких може відображатися на будь-яку область фізичної пам'яті. Починаючи з 5-го покоління, з'явилася можливість збільшення розміру сторінки до 4 Мбайт. Сегментація і розбивка на сторінки можуть застосовуватися в будь-яких сполученнях. Сегментація є засобом організації логічної пам'яті, використовуваним на прикладному рівні. Розбивка на сторінки застосовується на системному рівні для керування фізичною пам'яттю.

Стосовно до пам'яті розрізняють три адресних простори: логічне, лінійне і фізичне. Основним режимом роботи 32-розрядних процесорів вважається захищений режим, у якому працюють усі механізми перетворення адресних просторів.

Логічна адреса, також називана віртуальною, складається із селектора сегмента (у реальному режимі - просто адреси сегмента) і ефективної адреси, називаного також зсувом (offset). Ефективна адреса формується підсумовуванням компонентів base, index, displacement з урахуванням масштабу scale. Оскільки

кожна задача може мати до 16 К селекторів, а зсув, обмежений розміром сегмента, може досягати 4 Гбайт, логічний адресний простір для кожної задачі може досягати 64 Тбайт. Весь цій простір віртуальної пам'яті в принципі доступний програмісту (за умовою підтримки з боку операційної системи).

Блок сегментації транслює логічний адресний простір у 32-бітний простір лінійних адрес. *Лінійна адреса* утворюється додаванням базової адреси сегмента з ефективною адресою. Базова адреса сегмента в реальному режимі утворюється множенням вмісту використовуваного сегментного регістра на 16 (як і в 8086). У захищеному режимі базова адреса завантажується з дескриптора, що зберігається в таблиці, по селектору, завантаженому у використовуваний сегментний регістр.

32-бітна фізична адреса пам'яті утвориться після перетворення лінійної адреси блоком сторінкової переадресації. Він виводиться на зовнішню шину адреси процесора. У найпростішому випадку (при відключеному блоці сторінкової переадресації) фізична адреса збігається з лінійним. Включений блок сторінкової переадресації здійснює трансляцію лінійної адреси у фізичну сторінками розміром 4 Кбайт (для старших поколінь процесорів також можливі сторінки розміром 2 чи 4 Мбайт). Блок забезпечує розширення розрядності фізичної адреси процесорів шостого покоління до 36 біт. Блок переадресації може включатися тільки в захищеному режимі.

Для звертання до пам'яті процесор (разом із зовнішньою схемою) формує шинні сигнали MEMWR# (Memory Write) і MEMRD# (Memory Read) для операцій запису і читання відповідно. Шина адреси розрядністю 32 біта дозволяє адресувати 4 Гбайт фізичної пам'яті, але в реальному режимі доступний тільки 1 Мбайт, що починається з молодших адрес.

У реальному режимі адресації пам'яті забезпечується сумісність із процесором 8086, що своєю 16-розрядною адресною шиною охоплює простір фізичної пам'яті в 1 Мбайт. У реальному режимі розмір сегмента фіксований - як і в 8086, він складає 64 Кбайт (FFFFh). Спроба використання ефективної адреси,

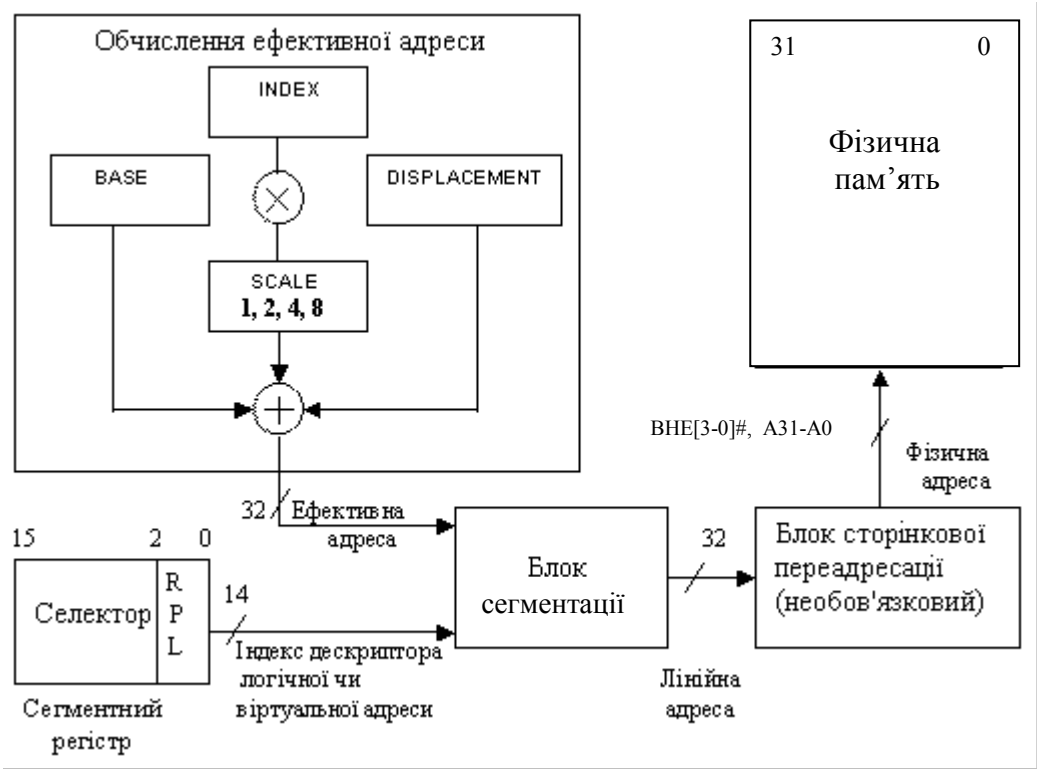

Рисунок 1 - Формування фізичної адреси

що виходить за межу сегмента, при 32-бітній адресації викликає виключення #GP. При 16-бітній адресації при обчисленні ефективної адреси можливий перенос у розряд A16 ігнорується і сегмент "звертається в кільце" (як і в 8086). Засоби контролю стежать і за переходом через границю сегмента під час звертання по "приграничній" адресі. При спробі адресації до слова, що має зсув FFFFh, чи подвійному слову зі зсувом FFFDh-FFFFh (їхні старші байти виходять за межу сегмента), чи виконанні інструкції, хоча б один байт якої не уміщається в даному сегменті, процесор виробляє виключення #GP. При спробі виконання інструкції співпроцесора (ESCAPE) з операндом пам'яті, що не уміщається в сегменті, виробляється виключення 9 - Processor Extension Segment Overrrun Interrupt (тільки для 80386).

Система команд 32-розрядних процесорів передбачає 11 режимів адресації. При цьому тільки в двох випадках операнди не зв'язані з пам'яттю. Це операнд-вміст регістра, що береться з кожного 8-, 16- чи 32-бітного регістра процесора, і

безпосередній операнд (8, 16 чи 32 біт), що міститься в самій команді. Інші дев'ять режимів так чи інакше звертаються до пам'яті.

При звертанні до пам'яті ефективна адреса обчислюється з використанням наступних компонентів:

- *Зсув* (Displacement або Disp) - 8-, 16- чи 32-бітне число, включене у команду.
- *База* (Base) - вміст базового регістра. Звичайно використовується для вказівки на початок деякого масиву.
- *Індекс* (Index) - вміст індексного регістра. Звичайно використовується для вибору елемента масиву,
- *Масштаб* (Scale) - множник (1, 2, 4 чи 8), зазначений у коді інструкції. Цей елемент використовується для вказівки розміру елемента масиву, доступний тільки при 32-бітній адресації. Ефективна адреса обчислюється по формулі

EA = Base + Index x Scale + Disp

Окремі доданки в цій формулі можуть бути відсутні.

2. Захищений режим

Захищений режим *Protected Mode*, більш урочисто називаний *Protected Virtual Address Mode* (захищений режим віртуальної адресації), є основним (найбільш природним) режимом роботи 32-розрядних процесорів. У цьому режимі процесор дозволяє адресувати до 4 Гбайт (до 64 Гбайт у P6) фізичної пам'яті, через які при використанні механізму сегментації можуть відображатися до 64 Тбайт віртуальної пам'яті кожної задачі. Режим віртуального процесора 8086 - *Virtual 8086 Mode* або *V86* - є особливим станом задачі захищеного режиму, у якому процесор функціонує як 8086 з можливістю використання 32-розрядних адрес і операндів.

Захищений режим з'явився ще в процесорі 80286, але мав не всі можливості, доступні в 32-розрядних процесорах.

2.1 Основні поняття захищеного режиму

Захищений режим призначений для забезпечення незалежності виконання декількох задач, що має на увазі захист ресурсів однієї задачі від можливого впливу іншої (під задачами припускаються як додатки, так і задачі операційної системи).

Основним ресурсом, що захищається, є пам'ять, у якій зберігаються коди, дані і різні системні таблиці (наприклад, таблиця переривань). Захищати потрібно і спільно використовувану апаратуру, звертання до якої звичайно відбувається через операції в/в і переривання. У захищеному режимі процесор апаратно реалізує багато функцій захисту, необхідні для побудови супервізора багатозадачної ОС, у тому числі механізм віртуальної пам'яті.

Захист пам'яті заснований на *сегментації. Сегмент* - це блок простору пам'яті певного призначення. До елементів сегмента можливе звертання за допомогою різних інструкцій процесора, що використовують різні режими адресації для формування адреси в межах сегмента. Максимальний розмір сегмента - 4 Гбайт (для процесорів 8086 і 80286 межа була всього 64 Кбайт).

Сегменти пам'яті виділяються задачам операційною системою, але в реальному режимі будь-яка задача може перевизначити значення сегментних регістрів, що задають положення сегмента в просторі пам'яті, і потрапити у чужу область даних або коду. У захищеному режимі сегменти теж розподіляються операційною системою, але прикладна програма зможе використовувати тільки дозволені для неї сегменти пам'яті, вибираючи їх за допомогою *селекторів* з попередньо сформованих *таблиць дескрипторів сегментів*.

Процесор може звертатися тільки до тих сегментів пам'яті, для яких є дескриптори в таблицях. Механізм сегментації формує лінійну адресу за схемою, приведеною на рисунку 2.

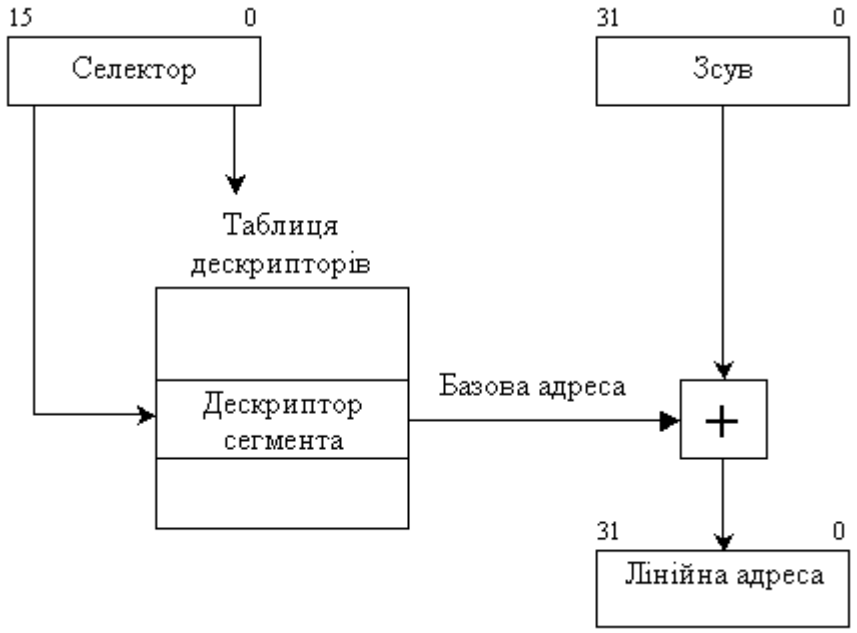

Рисунок 2 - Формування лінійної адреси в захищеному режимі

Дескриптори вибираються за допомогою 16-бітних селекторів. Формат селекторів, що завантажуються програмно в сегментні регістри, приведений на рисунку 3.

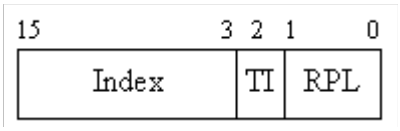

Рисунок 3 - Формат селектора

Індекс разом з індикатором таблиці TI дозволяє вибрати дескриптор з локальної (TI =1) чи глобальної (TI =0) таблиці дескрипторів. Для невикористовуваних сегментних регістрів призначений нульовий селектор сегмента, що формально адресується до найпершого елементу глобальної таблиці. Спроба звертання до пам'яті по такому сегментному регістру викликає виключення. Виключення виникне і при спробі завантаження нульового селектора в регістр CS чи SS. Поле RPL указує необхідний рівень привілеїв.

Дескриптори являють собою 8-байтні структури даних, використовувані для визначення властивостей програмних елементів (сегментів, вентилів і таблиць). Дескриптор визначає положення елемента в пам'яті, розмір займаної їм області (ліміт), його призначення і характеристики захисту. Усі дескриптори зберігаються в таблицях, звертання до яких підтримується процесором апаратно.

Захист пам'яті за допомогою сегментації не дозволяє:

- використовувати сегменти не по призначенню (наприклад, намагатися трактувати область даних як коди інструкцій);
- порушувати права доступу (намагатися модифікувати сегмент, призначений тільки для читання, звертатися до сегмента, не маючи достатніх привілеїв тощо);
- адресуватися до елементів, що виходять за ліміт сегмента;
- змінювати вміст таблиць дескрипторів (тобто параметрів сегментів), не маючи достатніх привілеїв.

Захищений режим надає засоби *переключення задач*. Стан кожної задачі (значення всіх зв'язаних з нею регістрів процесора) може бути збережений в спеціальному сегменті стану задачі *TSS*, на який указує селектор у регістрі задачі *TR*. При переключенні задач досить завантажити новий селектор у регістр задачі, і стан поточної задачі автоматично збережеться в її *TSS*, а в процесор завантажиться стан нової (можливо, раніше перерваної) задачі, і почнеться (продовжиться) її виконання.

Чотирьохрівнева ієрархічна система привілеїв призначена для керування використанням привілейованих інструкцій і доступом до дескрипторів. Рівні привілеїв нумеруються від 0 до 3, нульовий рівень відповідає максимальним (необмеженим) можливостям доступу і приділяється для ядра операційної системи. Рівень 3 має самі обмежені права і звичайно надається прикладним задачам. Систему захисту звичайно зображують у виді концентричних кілець, що відповідають рівням привілеїв (малюнок), а самі рівні привілеїв іноді називають кільцями захисту. Сервіси, надані задачам, можуть знаходитися в різних кільцях

захисту. Передача керування між задачами контролюється *вентилями* (Gate), називаними також *шлюзами*, що перевіряють правила використання рівнів привілеїв. Через вентилі задачі можуть одержати доступ тільки до дозволених їм сервісам інших сегментів.

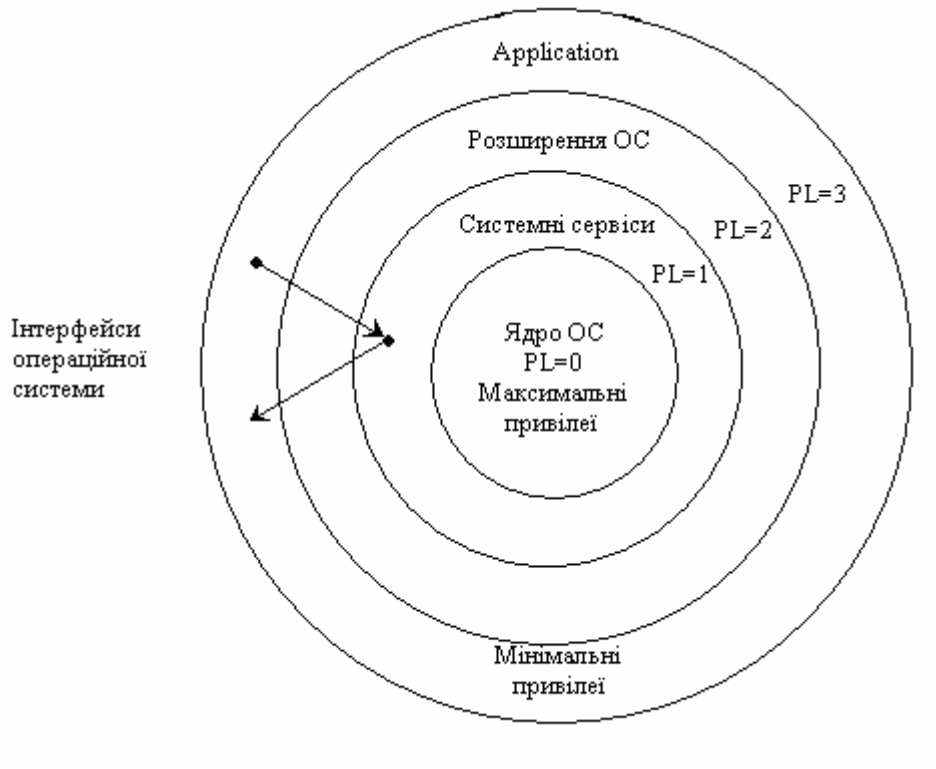

Рисунок 4 - Рівні привілеїв

Рівні привілеїв відносяться до дескрипторів, селекторів і задач. Крім того, у регістрі прапорів є *поле привілеїв в/в*, за допомогою якого забезпечується керування доступом до інструкцій в/в і керуванням прапором переривань.

Дескриптори і привілеї є основою системи захисту: дескриптори визначають структури програмних елементів (без яких ці елементи неможливо використовувати), а привілеї визначають можливість доступу до дескрипторів і виконання привілейованих інструкцій. Будь-яке порушення захисту приводить до виникнення спеціальних виключень, оброблюваних ядром операційної системи.

Механізм віртуальної пам'яті дозволяє будь-якій задачі використовувати логічний адресний простір розміром до 64 Тбайт (16К сегментів по 4 Гбайт). Для цього кожен сегмент у своєму дескрипторі має спеціальний біт, що вказує на

присутність даного сегмента в оперативній пам'яті в даний момент часу. Невикористовуваний сегмент може бути вивантажений з оперативної пам'яті в зовнішню (наприклад, дискову), про що робиться позначка в його дескрипторі. На місце, що звільнилося, із зовнішньої пам'яті може відновлюватися вміст іншого сегмента (цей процес називається свопінгом чи підкачуванням), і в його дескрипторі робиться позначка про присутність у пам'яті. При звертанні задачі до відсутнього сегмента процесор виробляє відповідне виключення, оброблювач якого і завідує віртуальною пам'яттю в операційній системі. Механізм сторінкової переадресації забезпечує віртуалізацію пам'яті, адресуємою логічною адресою, на рівні сторінок фіксованого розміру. Після підкачування сегмента (сторінки) виконання задачі продовжується, так що віртуалізація пам'яті для прикладних задач прозора (якщо не брати до уваги затримку, викликану підкачуванням).

Процесор надає тільки необхідні апаратні засоби підтримки захисту і віртуальної пам'яті, а їхнє реальне використання і стійкість роботи програм і самої операційної системи захищеного режиму, звичайно ж, залежать від коректності побудови ОС і передбачливості її розроблювачів. Добре спроектована операційна система захищеного режиму може забезпечити стійкість ОС навіть при некоректному поводженні прикладних задач.

2.2 Дескриптори і таблиці

Існують три типи таблиць дескрипторів - *локальна таблиця дескрипторів LDT* (Local Descriptor Table), *глобальна таблиця дескрипторів GDT* (Global Descriptor Table) і *таблиця дескрипторів переривань IDT* (Interrupt Descriptor Table). Розміри таблиць можуть знаходитися в межах 8 байт - 64 Кбайт, що відповідає числу елементів у таблиці від 1 до 8К.

З кожною з цих таблиць зв'язаний відповідний регістр процесора. Регістри GDTR і IDTR мають програмно-доступне 16-бітне поле ліміту, що задає розмір таблиці, і 32-бітне (у 80286 - 24-бітне) поле базової адреси, що визначає

положення таблиці в просторі лінійних (у 80286 - фізичних) адрес пам'яті. У регістрі LDTR програмно доступно тільки 16-бітне поле селектора, по якому з LDT автоматично завантажуються програмно недоступні і невидимі поля базової адреси і ліміту.

Глобальна таблиця (GDT) містить дескриптори, доступні всім задачам. Вона може містити дескриптори будь-яких типів, крім дескрипторів переривань і пасток. Нульовий елемент цієї таблиці процесором не використовується. Локальна таблиця (LDT) може бути власною для кожної задачі і містить тільки дескриптори сегментів, вентилів задач і викликів. Сегмент недоступний задачі, якщо його дескриптора немає в даний момент ні в GDT, ні в LDT.

Вибір таблиці (локальна чи глобальна) визначається за значенням біта T1 селектора, а положення (номер) дескриптора задається 13-бітним полем INDEX селектора. При посиланні на дескриптор, що виходить за ліміт таблиці, виникає виключення #GP.

Таблиця дескрипторів переривань, використовувана в захищеному режимі, може містити опис до 256 переривань. Таблиця може містити тільки вентилі задач, переривань і пасток. Базова адреса і ліміт таблиці завантажується привілейованою командою LIDT (аналогічно LGDT). Розмір IDT повинний бути не менш 256 байт, для того щоб у неї помістилися всі зарезервовані переривання процесора. Посилання на елементи IDT відбуваються по командах INT, апаратним перериванням і виключенням процесора. При виникненні переривання виключення, дескриптор якого виходить за ліміт таблиці, виробляється виключення #DF.

Дескриптори мають 8-байтний формат як для 16-розрядних (80286), так і для 32-розрядних процесорів. Призначення дескриптора визначається полями байта керування доступом (Access Rights Byte) - байта зі зсувом 5. Дескриптори 16- і 32-розрядних процесорів відрізняються розрядністю полей базової адреси (24 і 32 біт) і трактуванням поля ліміту, що повинне забезпечувати розмір сегмента до 64 Кбайт чи 4 Гбайт відповідно. Два старших байти у дескрипторів 80286 завжди

нульові (через вимоги сумісності з наступними процесорами, оголошеними при випуску 80286), що дозволяє їх відрізняти і коректно використовувати, виконуючи 16-бітні програми захищеного режиму на 32-розрядних процесорах.

Два старших байти дескрипторів 32-розрядних процесорів містять розширення полів BASE і LIMIT, біт дробності G (Granularity), що визначає, у яких одиницях заданий ліміт: G=0 - у байтах, G=1 - у сторінках по 4 Кбайт (що і забезпечує максимальну довжину в 4 Гбайт).

У дескрипторах сегментів, відсутніх у фізичній пам'яті (P=0), процесор цікавиться тільки байтом керування доступом. Інші байти можуть використовуватися за розсудом ОС.

2.2.1 Дескриптори сегментів коду і даних

Дескриптори сегментів коду і даних визначають базові адреси, розмір сегмента, права доступу (читання, читання/запис, тільки виконання коду чи виконання/читання), а для систем з віртуальною пам'яттю ще і присутність сегмента у фізичній пам'яті.

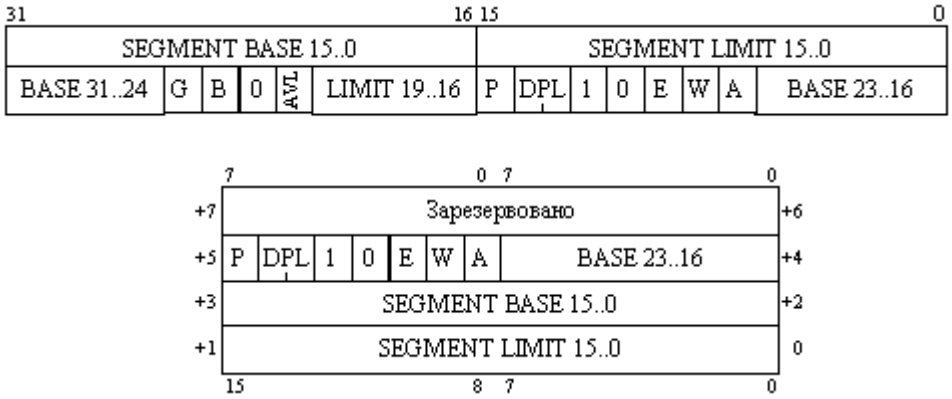

Рисунок 5 - Дескриптор сегмента даних: а – 32-бітний формат;
б – 16-бітний формат у стилі 80286.

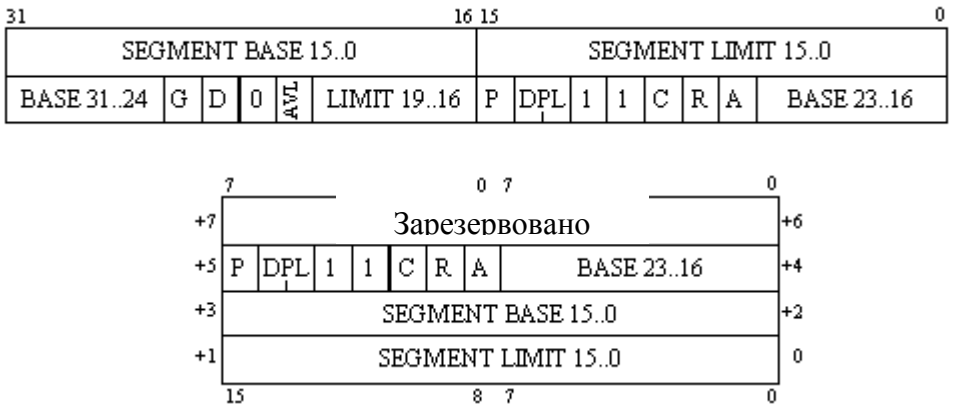

Рисунок 6 - Дескриптор сегмента коду: а – 32-бітний формат;
б – 16-бітний формат у стилі 80286.

У байті керування доступом поля мають наступне призначення:

- Біт 7 - P (Present) - присутність у пам'яті. При P=1 сегмент відображений у фізичній пам'яті, при P=0 відображення немає і поля базової адреси і ліміту не використовуються.
- Біти 6, 5 - DPL (Descriptor Privilege Level) - атрибути привілеїв сегмента.
- Біт 0 - A (Accessed) - звертання. A=0 - до сегмента не було звертання, A=1 - селектор даного сегмента завантажувався в регістр сегменту чи для нього виконувалася команда тестування.

Для дескриптора сегмента даних (включаючи і стек):

- Біт 2 - E (Expand Down) - контрольований напрямок розширення: E=0 - розширюваний нагору (зсув не повинний перевищувати значення ліміту, E=1 - розширюваний униз (стек, у якого зсув повинний перевищувати значення ліміту).
- Біт 1 - W (Writeable) - дозвіл (W=1) чи заборона (W=0) запису даних у сегмент.

Для сегментів даних (включаючи і стек), розширюваних униз (E=1), біт B в передостанньому байті дескриптора визначає верхню границю сегмента: при B=0

- FFFFh (максимальний розмір сегмента - 64 Кбайт), при B=1 - FFFFFFFFh (максимальний розмір сегмента - 4 Гбайт). Цей же біт у дескрипторі сегмента стека визначає і розрядність використовуваного покажчика стека: B=0 - використовується 16-бітний SP, розрядність даних при операціях PUSH, POP -16 біт; при B=1 використовується 32-бітний ESP, розрядність даних при операціях PUSH, POP- 32 біт.

У сегмент коду запис неможливий, ліміт указує на його останній байт, а біти типу мають наступне призначення:

- Біт 2 - C (Conforming, конформність чи підпорядкованість): при C=1 код може виповнюватися, якщо поточний рівень привілеїв (CPL) не нижче рівня привілеїв дескриптора (DPL); при C=0 (непідлеглий сегмент) керування до даного сегмента може передаватися, тільки якщо CPL=DPL.
- Біт 1 - R (Readable) - дозвіл (R=1) чи заборона (R=0) читання сегмента.

Запис у сегмент коду можлива тільки через псевдонім (Alias) - сегмент даних з дозволеним записом, що має те ж значення бази і ліміту.

Біт D (Default Operation Size) у передостанньому байті дескриптора сегмента коду визначає розрядність адрес і операндів за замовчуванням: D=0 - 16 біт, D=1 - 32 біт.

Тепер, коли ми знаємо про три ключові частини у визначенні адреси, давайте розглянемо, як це відбувається (припустимо, що проблем із захистом поки немає). У сегментний регістр завантажене значення селектора. Значення селектора розбите на три частини: (індекс, TI і RPL). Дескриптор вибирається з відповідної таблиці дескрипторів за допомогою TI і індекса (RPL використовується для перевірки наявності захисту). Вміст дескриптора завантажений у внутрішній дескриптор, відведений для відповідного сегментного регістра. Надалі, коли б ми не зверталися до пам'яті, використовуючи даний

сегментний регістр, потрібно додавати зсув до фізичної адреси у внутрішньому дескрипторі.

Приклад:

Mov AX, 18h

Mov DS, AX

Mov ESI, 22h

Mov EAX, [ESI]: потрібний зсув

1) Розпишемо значення селектора: 18h = 0000000000011 0 00

Індекс = 3

Індикатор таблиці (TI) указує, що дескриптор знаходиться в GDT.

Рівень привілеїв запитуючої сторони (RPL) не використовується в обчисленні адреси.

2) Знайти індексований дескриптор у відповідній таблиці дескрипторів, TI= 0, виходить, адресуємося до GDT

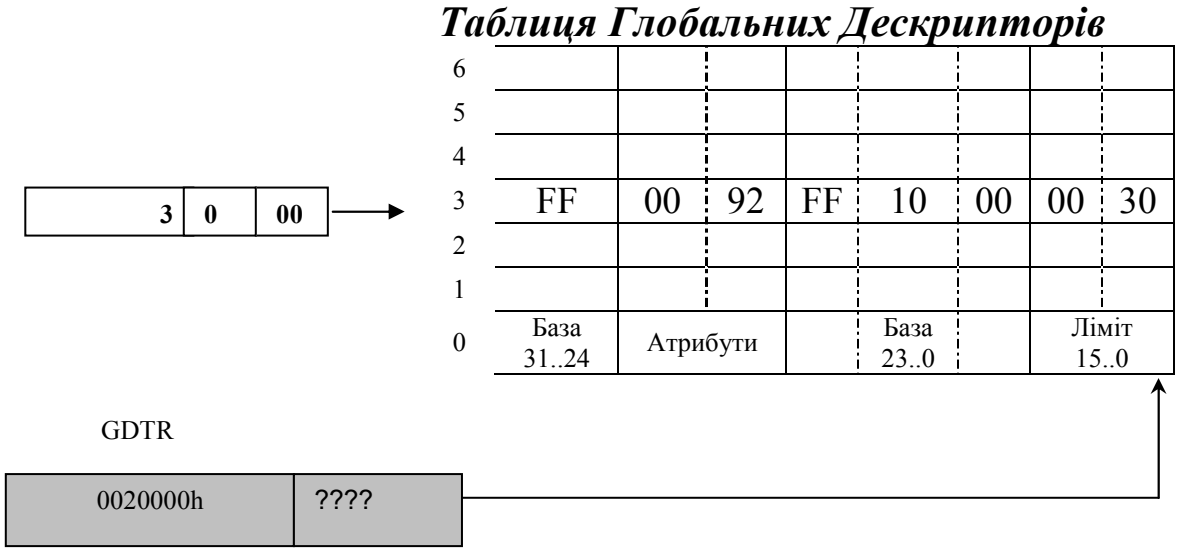

Рисунок 7 – Приклад визначення фізичної адреси

3) Витягти базову адресу з обраного дескриптора: 0FFFF1000h.

4) Додати зсув до бази: ESI = 22h, 0FFFF1000h + 22h = 0FFFF1022h - необхідна адреса.

Ці структури системних даних (селектори, дескриптори, і таблиці дескрипторів) є основними при операціях у захищеному режимі.

2.2.2 Дескриптор системних сегментів і дескриптори вентилів

Системні сегменти призначені для збереження локальних таблиць дескрипторів LDT (Local Descriptor Table) і стану задач TSS (Task State Segment). Їхні дескриптори визначають базову адресу, ліміт сегмента (1 – 64 Кбайт), права доступу (читання, читання/запис, тільки виконання коду чи виконання/читання) і присутність сегмента у фізичній пам'яті.

У байті керування доступний у цих дескрипторів біт P визначає дійсність (P=1) чи недійсність (P=0) вмісту сегмента. Поле рівня привілеїв DPL використовується тільки у дескрипторах сегментів стану задач. Оскільки звертання до локальних дескрипторів можливо тільки по привілейованих командах, поле DPL для дескрипторів таблиць не використовується. Поле Type (1-3, 9-B) визначає тип сегмента:

- 0,8 - неприпустимі значення;
- 1 - доступний сегмент стану задачі 80286 (Available TSS-286);
- 2 - таблиця локальних дескрипторів (LDT);
- 3 - зайнятий сегмент стану задачі 80286 (Busy TSS-286);
- 9 - доступний сегмент стану задачі 386+ (Available TSS-386);
- A - не визначено (зарезервовано);
- B - зайнятий сегмент стану задачі 386+

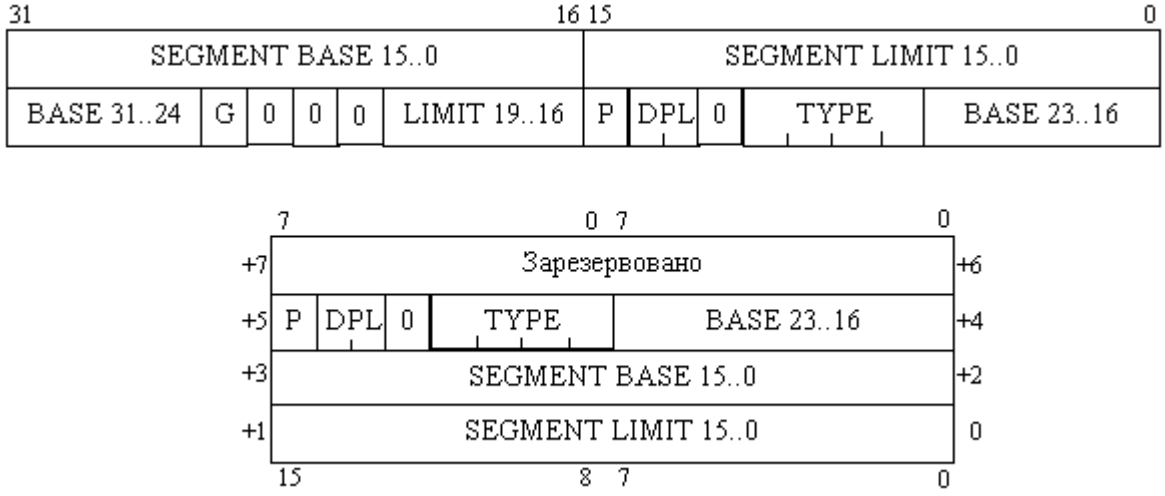

Рисунок 8 - Дескриптор системних сегментів: а – 32-бітний формат;
б – 16-бітний формат 80286.

Межсегментна передача керування безпосередньо (командами JMP, CALL, INT, RET і IRET) можлива тільки до сегментів коду з тим же рівнем привілеїв або до підлеглих сегментів, рівень привілеїв яких вище, ніж CPL (при цьому CPL не змінюється). Для переходів зі зміною рівня привілеїв використовуються *вентилі* (Gate), іноді називані шлюзами. Для кожного способу непрямої межсегментної передачі керування є відповідні вентилі, їхнє використання дозволяє процесору автоматично виконувати контроль захисту. *Вентилі виклику* (Call Gates) використовуються для викликів процедур зі зміною рівня привілеїв, *вентилі задач* (Task Gates) використовуються для переключення задач, *вентилі переривань* (Interrupt Gates) і *пасток* (Trap Gates) визначають процедури обслуговування переривань. Вентилі виклику дозволяють автоматично копіювати задане число слів зі старого стека в новий. Вентилі переривань відрізняються від вентилів пасток тільки тим, що вони забороняють переривання (скидають IF), а вентилі пасток - ні. Для кожного типу вентилів використовуються відповідні *дескриптори вентилів* (Gate Descriptors).

У байті керування доступний у цих дескрипторів біт P визначає дійсність (P=1) чи недійсність (P=0) вмісту сегмента. Поле DPL задає рівень привілеїв. Поле *Type* визначає тип вентиля:

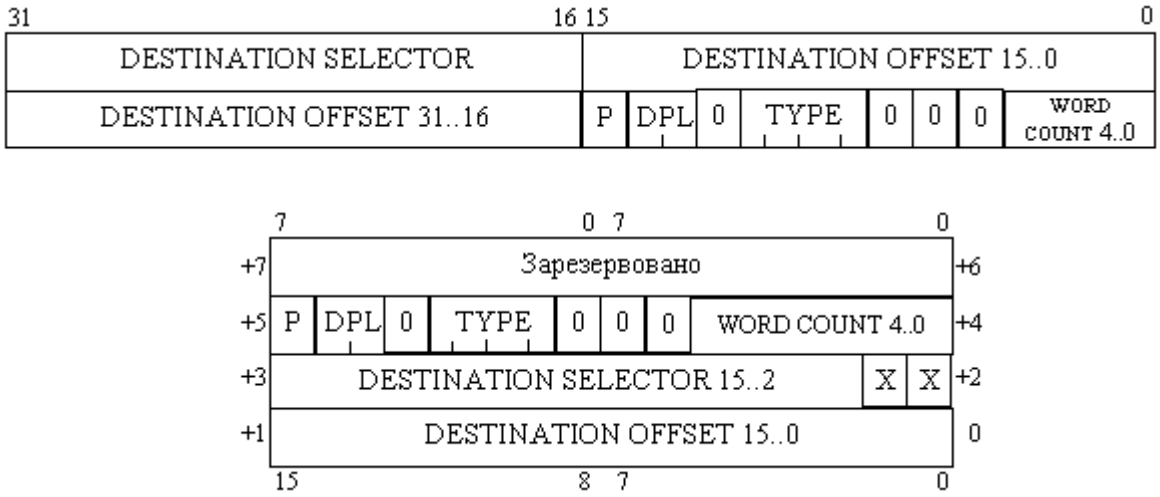

Рисунок 9 - Дескриптори вентилів: а – 32-бітний формат;
б – 16-бітний формат у стилі 80286.

- 4 - вентиль виклику 80286 (Call Gate);
- 5 - вентиль задачі 80286 (Task Gate);
- 6 - вентиль переривання 80286 (Interrupt Gate);
- 7 - вентиль пастки 80286 (Trap Gate);
- C - вентиль виклику 386+ (Call Gate);
- D - вентиль задачі 386+ (Task Gate);
- E - вентиль переривання 386+ (Interrupt Gate);
- F - вентиль пастки 386+ (Trap Gate).

Поле *Word Count* використовується тільки у вентилях викликів і визначає число слів зі стека процесу, що викликається і що автоматично копіюється у стек викликуваної процедури. Для сегментів 80286 слова 16-бітні, для 386+ - 32-бітні.

Слово *Destination Selector* для вентилів виклику, переривань і пасток задає селектор цільового сегмента коду, а для вентиля задачі - селектор цільового TSS.

Слово *Destination Offset* задає зсув (адреса) точки входу в цільовому сегменті.

При використанні вентилів може виникнути виключення #GP, що означає, що селектор указує на некоректний тип дескриптора. При спробі використання недійсного вентиля (P=0) виникає виключення #NP.

2.3 Вентилі запиту

Вентиль запиту - метод, що використовується для передачі керування від коду з більш низькими привілеями коду з більш високими привілеями. Дескриптор вентиля має чотири поля: значення селектора адресата, 32-розрядний зсув на приймач, індекс копії параметра слова і DPL. Селектор адресата в дескрипторі вентиля запиту - селектор для сегмента коду, що буде адресатом CALL або JMP. 32-розрядний зсув - зсув у сегменті адресата (те, що буде завантажуватися в EIP). Слово параметрів - та кількість інформації, що повинна бути скопійована зі стека з більш низьким пріоритетом у стек з більш високим пріоритетом. Вентиль DPL, подібно погодженому сегменту DPL, використовується, щоб обмовити, які рівні привілеїв мають доступ до вентиля. Наприклад, до вентиля з DPL 2 можна звертатися кодами, що виконуються в рівнях привілеїв 0, 1, чи 2, але якщо звертатися кодом з рівнем привілеїв 3, відбудеться загальна помилка захисту (general protection fault).

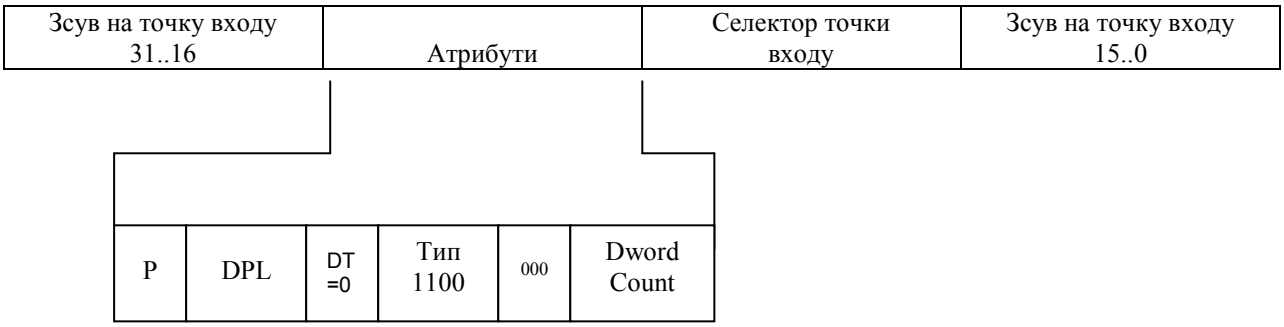

Рисунок 10 - Формат дескриптора вентиля виклику

При завантаженні селектора в CS для одержання дескриптора вентиля запиту (дескриптор системного типу, а не сегмент), вентиль DPL повинний

дорівнювати чи бути більшим чим максимум CPL і RPL. CPL повинний бути більше чи дорівнювати адресату DPL

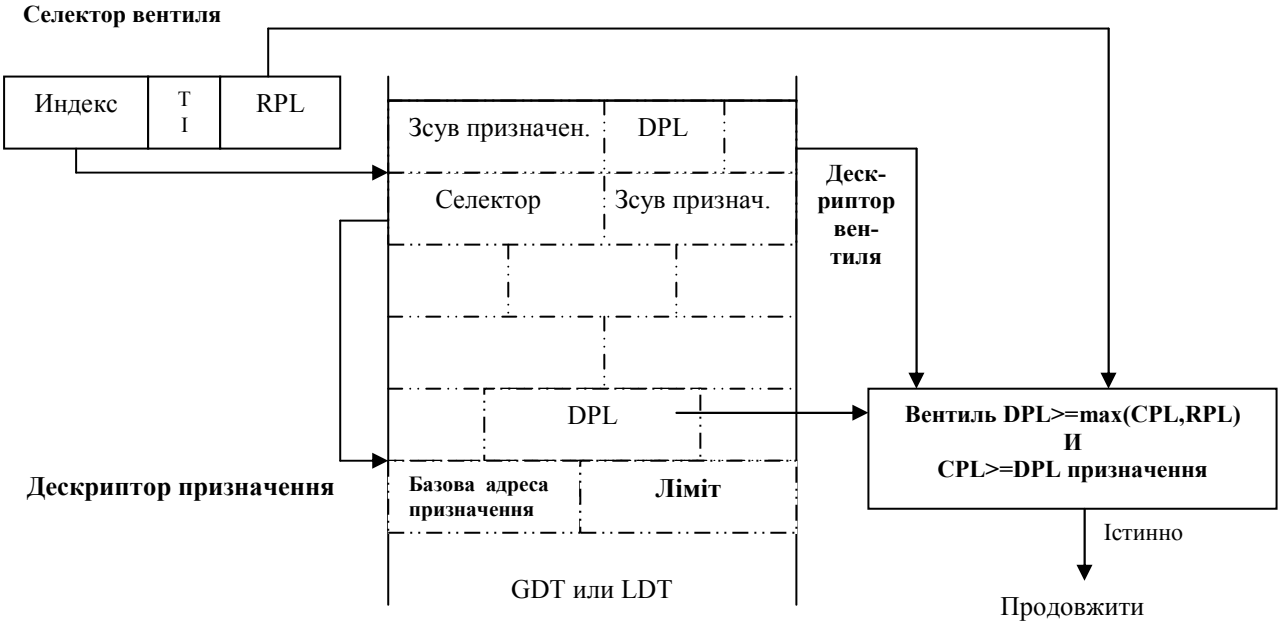

Рисунок 11 - Перевірка привілеїв вентиля виклику

Таким чином, якщо треба викликати код або перейти до коду з більш високим рівнем привілеїв, треба адресуватися через вентиль.

2.4 Переривання

Процес переривання відбувається приблизно так, як і в 8086. Виключенням є: таблиці нових значень CS:IP, таблиця задачі, вентилі чи переривання пастки (відомі, як дескриптор таблиці переривань (IDT)).

На IDT указує регістр IDTR, що містить базову адресу IDT і ліміт. Тому що переривання можуть відбуватися в будь-який момент часу, вентилі використовуються для гарантій того, що передача керування буде виконана незалежно від CPL. Вентилі переривань і пасток працюють так само, як і клапани запиту без передачі параметрів від одного стека рівня привілеїв іншому. Вентилі переривань і пасток при виконанні програми "поводяться" однаково, за винятком

того, що для установки переривання зводиться прапор (IF) у регістрі EFLAGS. Вентилі переривань скидають IF прапор, відключаючи тим самим переривання доти, поки вентиль пастки не змінить його. Потім EFLAGS завантажується в стек, як частина процесу переривання, стан прапора IF відновлюється, як тільки виконується команда IRET. Перериваючи це посилання, вентиль викликає переключення задач.

Джерелами переривань є: зовнішні події, виключення, і команди (INT0, INT3, INT n, і BOUND). Інформація, що поміщається в стек для джерел переривання зовнішньої події чи команд - однакова, у той час як для виключень процесора може бути поміщена в стек додаткова інформація (код помилки).

Розглянемо приклад: процесор виконує інструкцію INT 2 (вектор переривання процесу):

· Обчислюється адреса, за якою береться дескриптор переривання. Номер_Вектора * Розмір_Дескриптора (2*8) + IDTR (отримана величина і є фізичною адресою дескриптора переривання).

· Зчитується дескриптор для визначення сегмента і зсуву усередині сегмента на вектор переривання. Наш дескриптор показує 18h для селектора, що представляє номер 3 входу таблиці GDT і 00003B2Ch як зсув у даному сегменті. 8Fh це атрибути, F повідомляє нам, що присутній вентиль пастки.

· Зчитується дескриптор з GDT[3] для визначення фізичної адреси призначення і складається зі зсувом у IDT[2] вентиля пастки для одержання лінійної адреси в пам'яті на вектор (0FF1000h + 3B2Ch = FF4B2Ch).

2.5 Привілеї

У захищеному режимі процесор має чотирьохрівневу систему привілеїв, що керує використанням привілейованих інструкцій і доступом до дескрипторів (і зв'язаним з ними сегментам). Рівні привілеїв нумеруються від 0 до 3, вищі

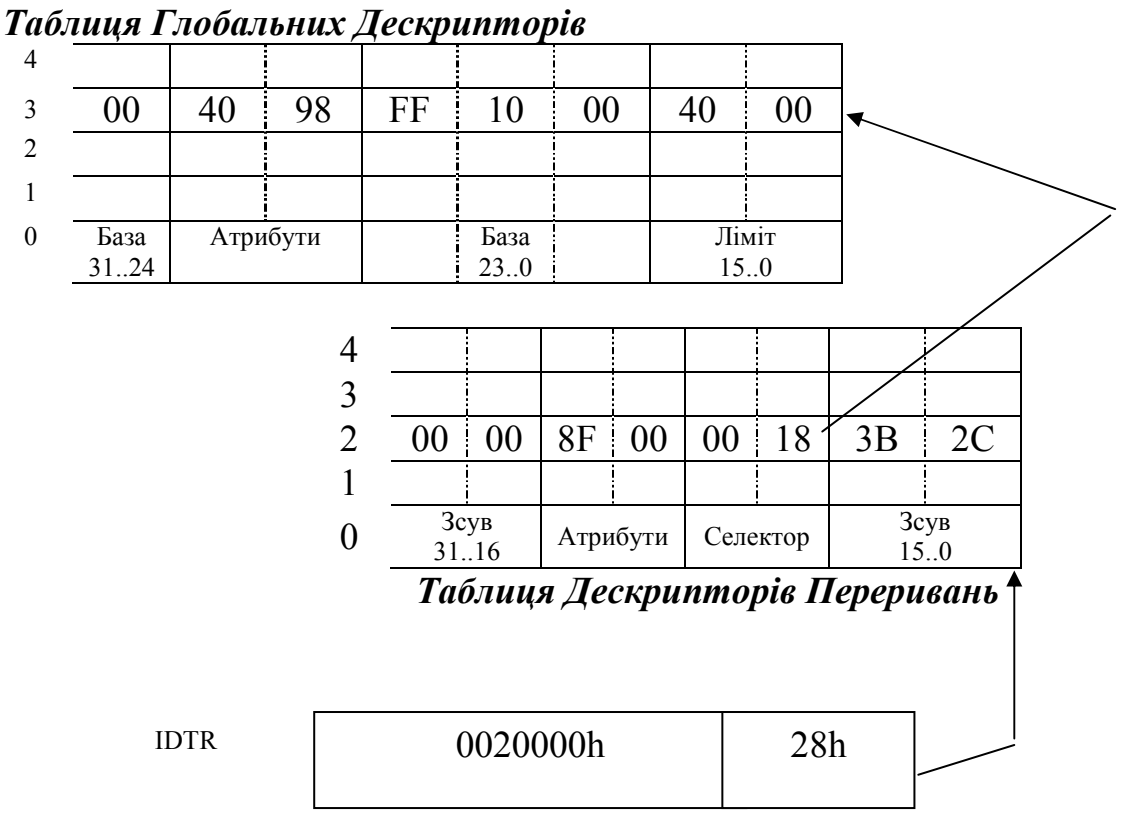

Рисунок 12 – Приклад визначення лінійної адреси програми обробки переривання

привілеї відповідають нульовому рівню. Рівні привілеїв забезпечують захист задач, изолюємих друг від друга локальними таблицями дескрипторів. Сервіс операційної системи, оброблювачі переривань і інше системне забезпечення можуть включатися у віртуальний адресний простір кожної задачі і захищатися системою привілеїв. Кожна частина системи працює на своєму рівні привілеїв. Задачі, дескриптори і селектори мають свої атрибути привілеїв.

Привілеї задач (Task Privilege) впливають на виконання інструкцій і використання дескрипторів. Поточний рівень привілеї задачі CPL (Current Privilege Level) визначається двома молодшими бітами регістра CS. CPL задачі може змінюватися тільки при передачі керування до нового сегмента через дескриптор вентиля. Задача починає виконуватися з рівня CPL, зазначеного селектором кодового сегмента усередині TSS, коли задача ініціюється за

допомогою операції переключення задач. Задача, виконувана на нульовому рівні привілеїв, має доступ до всіх сегментів, описаним у GDT, і є самою привілейованою. Задача, яка виконується на рівні 3, має найобмежені права доступу. Поточний рівень привілеї може змінюватися тільки при передачі керування через вентилі.

Привілеї дескриптора (Descriptor Privilege) задаються полем DPL байта керування доступом. DPL визначає найбільший номер рівня привілеїв (фактично, найменші привілеї), з яким можливий доступ до даного дескриптора. Самий захищений дескриптор має DPL=0, до нього мають доступ тільки задачі з CPL=0. Самий беззахисний дескриптор має DPL=3, його можуть використовувати задачі з CPL=0, 1, 2, 3. Це правило застосовне до всіх дескрипторів, за винятком дескриптора LDT.

Привілеї селектора (Selector Privilege) задаються полем RPL (Requested Privilege Level) - двома молодшими бітами селектора. За допомогою RPL можна урізати ефективний рівень привілеїв EPL (Effective Privilege Level), що визначається як максимальне зі значень CPL і RPL. Селектор з RPL=0 не вводить додаткових обмежень.

Контроль доступу до сегментів даних виробляється при виконанні команд, що завантажують селектори в SS, DS, ES, FS і GS. Команди завантаження DS, ES, FS і GS повинні посилатися на дескриптори сегментів даних чи сегментів кодів, що допускають читання. Для одержання доступу ефективний рівень привілеїв EPL повинний бути рівним чи меншим (арифметично) рівня привілеїв DPL дескриптора. Виключенням з цього правила є підлеглий сегмент коду, що читається, що може бути прочитаний задачею з будь-яким CPL. Якщо ефективний рівень привілеїв не дозволяє доступ чи посилання виробляється на некоректний тип дескриптора (на дескриптор вентиля чи на дескриптор кодового сегмента, що тільки виконується), виробляється виключення #GP. При посиланні на неіснуючий дескриптор виробляється виключення #NP.

Команди завантаження SS повинні посилатися на дескриптор сегмента даних, що допускає запис. При цьому DPL і PPL повинні бути рівні CPL. Порушення цієї умови і посилання на дескриптор іншого типу породжують виключення #GP, при посиланні на неіснуючий дескриптор виробляється виключення #SS.

Контроль типів і привілеїв при передачі керування виробляється при завантаженні селектора в регістр CS. Тип дескриптора, на який посилається даний селектор, повинний відповідати інструкції, яка виконується. Порушення типу (наприклад, посилання інструкції JMP на вентиль виклику) породжують виключення #GP. При передачі керування діють наступні правила привілеїв, порушення яких також приводить до виключення #GP:

- команди JMP чи CALL можуть посилатися або на підлеглий сегмент коду з DPL, більшим чи рівним CPL, або на непідлеглий сегмент із DPL рівним CPL;
- переривання усередині задачі або виклики, що можуть змінити рівень привілеїв, можуть передавати керування кодовому сегменту з рівнем привілеїв, рівним чи більшим рівня привілеїв CPL, тільки через вентилі з тим же чи меншим рівнем привілеїв, чим CPL;
- інструкції повернення, що не переключають задачі, можуть передати керування тільки кодовому сегменту з таким же чи меншим рівнем привілеїв;
- переключення задач може виконуватися за допомогою виклику, чи переходу переривання, що посилаються на вентиль задачі чи сегмент стану задачі (TSS) з тим же чи меншим рівнем привілеїв.

Зміна рівня привілеїв, що відбувається при передачі керування, автоматично викликає перевизначення стека. Початкове значення покажчика стека SS:SP для рівня привілеїв 0, 1, 2 міститься в TSS. При передачі керування по командах JMP або CALL у CS:SP завантажується нове значення покажчика стека, а старі значення містяться в новий стек. При поверненні на колишній рівень привілеїв його стек відновлюється (як частина інструкції RET чи IRET). Для

викликів підпрограм з передачею параметрів через стек і зміною рівня привілеїв з попереднього стека в новий копіюється фіксоване число слів, задане у вентилі. Команда міжсегментного повернення RET з вирівнюванням покажчика стека при поверненні коректно відновить значення попереднього покажчика.

3. Контрольні завдання

3.1 Завдання № 1

а) Визначити ефективну, лінійну, фізичну адреси в захищеному режимі віртуальної адресації 32-розрядного процесора з безсторінковою організацією пам'яті. Визначити і прокоментувати атрибути дескриптора. Вихідні дані:

Selector	Base	Index	Scale	Displacement
$8*N_{10} + N_{10}$ mod8	Annh	Bnnh	8	Cnnh

де N_{10} – номер завдання, nn – двозначний номер завдання. Global Descriptor Table– таблиця 1, Local Descriptor Table – таблиця 2.

б) При тих же значеннях Base, Index, Displacement визначити фізичну адресу в реальному режимі (режим віртуального процесора 8086). Значення сегментних регістрів рівні значенню селектора.

3.2 Завдання № 2

Визначити дескриптор і лінійну адресу виклику програми зі зміною рівня привілеїв. Виконати перевірку привілеїв вентиля виклику. Визначити варіанти значень CPL, DPL вентиля, RPL і DPL призначення (цільового сегмента коду), для яких не відбудеться загальної помилки захисту.

Вихідні дані: Селектор вентиля = 8*(N_{10}+30) + (N_{10}-1)mod8; CPL=(N_{10}-1)mod4; X= (N_{10}-1)mod3 +Ah, де N_{10} – номер завдання. Global Descriptor Table– таблиця 1, Local Descriptor Table – таблиця 2.

3.3 Завдання № 3

Визначити лінійну адресу програми обробки переривання, джерелом якого є команда INT n, де n=N_{10}, N – номер завдання. Визначити і прокоментувати атрибути дескриптора переривання і дескриптора глобальної або локальної таблиці. Вихідні дані: Global Descriptor Table– таблиця 1, Local Descriptor Table – таблиця 2, Interrupt Descriptor Table –таблиця 3.

4. Приклади

4.1 Завдання 1

Визначення варіанта: Група А, варіант 26.

а) Визначити ефективну, лінійну, фізичну адреси в захищеному режимі віртуальної адресації 32-розрядного процесора з безсторінковою організацією пам'яті. Визначити і прокоментувати атрибути дескриптора.

Вихідні дані:

Selector = 8*26 + 26 mod 8 = 210_{10}

Base = A26h

Index = B26h

Scale = 8

Displacement = C26h

Selector = 210_{10} = 11010010_2

15 3	2	1 0
Index	TI	RPL
0000000011010	0	10

TI = 0, значить дескриптор знаходиться в GDT.

Номер дескриптора – поле Index у селекторі: $11010_2 = 26_{10}$

Вибираємо 26-й дескриптор з GDT:

№	BaseSeg 31...24	Attribute		BaseSeg 23...0			Limit 15...0	
26	0F	1C	DC	FF	10	00	00	50

Поле BaseSeg у дескрипторі визначає початок сегмента.

BaseSeg = 0FFF1000h

Ефективна адреса:

EA = *Base* + *Index*Scale* + *Displacement* = A26h + B26h*8 + C26h = 6F7Ch

Лінійна адреса:

ЛА = BaseSeg + EA = 0FFF1000h + 6F7Ch = 0FFF7F7Ch

Фізична адреса збігається з лінійною, тому що використовується безсторінкова організація пам'яті:

ФА = ЛА = 0FFF7F7Ch

Коментар до атрибута дескриптора

Attribute = 1CDCh = 0001 1100 1101 1100_2

Ім'я біта	Значення	Визначення біта
G	0	Біт ступеня деталізації. Визначає, що одиниця виміру ліміту є байт.
D	0	Указує, що за замовчуванням використовуються 16-розрядні операнди й адресація.
AVL	1	Біт доступності. Цей біт використовується розроблювачем за своїм розсудом.
P	1	Біт присутності. Указує, що сегмент присутній в оперативній пам'яті.

DPL	10	Дескриптор рівня привілеїв. Використовується механізмом захисту.
DT	1	Тип дескриптора. Указує, що дескриптор описує сегмент.
C_0/D_0	1	Тип сегмента. Указує, що сегмент є сегментом коду.
C	1	Указує, що сегмент коду є підлеглим.
R	0	Указує, що читання із сегмента заборонено.
A	0	Біт звертання. Процесор автоматично встановлює цей біт при звертанні до сегмента. Біт скидається програмно. 0 означає, що до сегмента не було звертання.

б) При тих же значеннях Base, Index, Displacement визначити фізичну адресу в реальному режимі (режим віртуального процесора 8086). Значення сегментних регістрів рівні значенню селектора.

Вихідні дані:

SReg = 8*26 + 26 mod 8 = 210_{10} = $D2_{16}$

Base = A26h

Index = B26h

Displacement = C26h

Ефективна адреса:

EA = *Base* + *Index* + *Displacement* = A26h + B26h + C26h = 2172h

Фізична адреса:

ФА = Sreg*16 + EA = D2h*16 + 2172h = D20h + 2172h = 2E92h

4.2 Завдання 2

Визначення варіанта: Група А, варіант 26.

Вихідні дані:

Селектор вентиля = 8*(N+31) + Nmod8 - 8*(26+31) + 26mod8 - 456+2 = 458; CPL = Nmod4 = 26mod4 = 2; X = A.

1. Визначити дескриптор и лінійну адресу виклику програми зі зміною рівня привілеїв. Виконати перевірку привілеїв вентиля виклику.

Визначаємо дескриптор:

$458_{10} = 0000000111001\ 0\ 10_2$

Gate Selector

15 3	2	1 O
Index	TI	RPL
0000000111001	0	10

RPL=2

По значенню селектора визначаємо індекс - 57.

TI =0, значить індекс знаходиться в Global Descriptor Table:

	Offset 31. ..16		Attribute		Selector		Offset 15. ..0	
57	00	00	AC	03	00	20	OA	04

Offset = 00000A04h

Атрибути вентиля: AC03h = $1010\ 1100\ 0000\ 0011_2$

P	DPL	DT	Тип	000	Dword Count
1	01	0	1100	000	00011

DPL вентиля = 1

Селектор призначення = 0020h

15 3	2	1 O
Index	TI	RPL
0000000000100	0	00

По значенню селектора визначаємо індекс - 4.

TI =0, значить індекс знаходиться в Global Descriptor Table:

	Base 31. ..24	Attribute		Base 23. ..0			Limit 15. ..0	
04	0F	44	99	FF	10	00	00	50

Base = 0FFF1000h

Визначаємо лінійну адресу:

Лінійна адреса = *Base* + *Offset* = 0FFFl000h + 00000A04h = 0FFFlA04h

Визначаємо атрибути дескриптора: 4499h = 0100 0100 1001 1001h

7	6	5	4	3 0	7	6	5	4	3	2	1	0
G	D	0	AVL	Limit	P	DPL		DT	C_0	C	R	A
0	1	0	0	0100	1	0	0	1	1	0	0	1

DPL призначення = 0

Перевірка привілеїв:

DPL вентиля >= max (CPL, RPL)

1 max (2, 2) = 2

умова не виконується

CPL >= DPL призначення

2 0

умова виконується

Через те що одна із умов не виконується - відбувається **general protection fault.**

2. Визначити варіанти значень CPL, DPL вентиля, RPL і DPL призначення (цільового сегмента коду), для яких не відбудеться загальної помилки захисту.

У даному випадку, для уникнення загальної помилки захисту, достатньо щоб DPL вентиля був равний 2 або 3. При цьому виконується обидві умови перевірки привілеїв. Наприклад:

DPL вентиля >= max (CPL, RPL) 3 max (2, 2) = 2

умова виконується

CPL >= DPL призначення

2 0

умова виконується

Відбувається продовження обробки інструкцій.

4.3 Завдання 3

Визначення варіанта: Група А, варіант 26.

1. Визначити лінійну адресу програми обробки переривання, джерелом якого є команда INT n, де $n=N_{10}$, N - номер варіанта.

За варіантом визначаємо: INT 26. Отримуємо із IDT:

	Offset 31. ..16		Attribute		Selector		Offset 15. ..0	
26	00	00	8E	00	00	2C	0B	05

Offset = 00000A05h Селектор призначення = 002Ch

15 3 2 1 0

Index	TI	RPL
0000000000101	1	00

По значенню селектора визначаємо індекс - 5.

TI =1, значить індекс знаходиться в Local Descriptor Table:

	Base 31. ..24	Attribute		Base 23. ..0			Limit 15. ..0	
05	0F	1C	DC	FF	20	00	00	30

Base = 0FFF2000h

Визначаємо лінійну адресу:

Лінійна адреса = *Base* + *Offset* = 0FFF2000h + 00000B05h = 0FFF2B05h

2. Визначити и прокоментувати атрибути дескриптора переривання і дескриптора глобальної або локальної таблиці.

Атрибути дескриптора переривання:

8E00h =1000111000000000_2

P	DPL	0	Type	0	0	0	Word count
1	00	0	1110	0	0	0	00000

Коментарі атрибутів:

P	1	Визначає присутність сегмента у пам’яті
DPL	0	Задає рівень привілеїв
Type	Eh	Тип вениля: вентиль переривання 386+ (Interrupt Gate)
Word count	0	Використовується тільки в вентилях викликів і визначає число слів із стека процесу, що викликається і що автоматично копіюється у стек викликуваної процедури.

Атрибути дескриптора локальної таблиці:

1CDCh=0001 1100 1101 1100_2

Коментарі атрибутів дескриптора:

Им’я біта	Значення	Призначення біта
G	0	Біт ступеня деталізації. Визначає, що одиницею заданого ліміту є байт.
D	0	Визначає заданий за замовчуванням розмір операнда - 16 біт.
AVL	1	Біт доступності, цей біт використовується системним розроблювачем.
P	1	Біт присутності. Указує, що даний сегмент присутній в пам’яті. Може використовуватися, як помічник при використанні віртуальної системи пам’яті.
DPL (2-bits)	10	Дескриптор рівня привілеїв, використовується механізмом захисту.
DT	1	Тип дескриптора, Указує, що дескриптор є сегментним дескриптором.
C_0/D_0	1	Біт типу сегмента. Указує, що сегмент є сегментом кода.
C	1	Указує, що сегмент - сумісний (CONFORMING) сегмент кода
R	0	Визначає, що сегмент кода тільки виконуємий.
A	0	Біт звертання, процесор автоматично зводить цей біт всякий раз при звертанні до дескриптора. Біт очищається програмно. 0 означає, що до сегмента не було звертання.

Додаток

Таблиця Д1 - Global Descriptor Table

60	00	00	AC	00	00	0C	0X	01
59	00	00	CC	01	00	10	0X	02
58	00	00	EC	02	00	1C	0X	03
57	00	00	AC	03	00	20	0X	04
56	00	00	CC	04	00	2C	0X	05
55	00	00	EC	05	00	30	0X	06
54	00	00	AC	06	00	3C	0X	07
53	00	00	CC	07	00	40	0X	08
52	00	00	EC	08	00	4C	0X	09
51	00	00	AC	09	00	50	0X	10
50	00	00	CC	0A	00	5C	0X	11
49	00	00	EC	0B	00	60	0X	12
48	00	00	AC	0C	00	6C	0X	13
47	00	00	CC	0D	00	70	0X	14
46	00	00	EC	0E	00	7C	0X	15
45	00	00	AC	0F	00	80	0X	16
44	00	00	CC	10	00	8C	0X	17
43	00	00	EC	11	00	90	0X	18
42	00	00	AC	12	00	9C	0X	19
41	00	00	CC	13	00	A0	0X	20
40	00	00	EC	14	00	AC	0X	21
39	00	00	AC	15	00	B0	0X	22
38	00	00	CC	16	00	BC	0X	23
37	00	00	EC	17	00	C0	0X	24
36	00	00	AC	18	00	CC	0X	25
35	00	00	CC	19	00	D0	0X	26
34	00	00	EC	1A	00	DC	0X	27
33	00	00	AC	1B	00	E0	0X	28
32	00	00	CC	1C	00	EC	0X	29
31	00	00	EC	1D	00	F0	0X	30
	Offset 31…16		Attribute		Selector		Offset 15…0	

Таблиця Д1 - Global Descriptor Table (закінчення)

30	0F	91	DE	FF	10	00	00	50
29	0F	9F	B6	FF	10	00	00	50
28	0F	5E	9D	FF	10	00	00	50
27	0F	5D	F5	FF	10	00	00	50
26	0F	1C	DC	FF	10	00	00	50
25	0F	1B	B4	FF	10	00	00	50
24	0F	DA	9B	FF	10	00	00	50
23	0F	D9	F3	FF	10	00	00	50
22	0F	98	DA	FF	10	00	00	50
21	0F	97	B2	FF	10	00	00	50
20	0F	56	99	FF	10	00	00	50
19	0F	55	F1	FF	10	00	00	50
18	0F	14	D8	FF	10	00	00	50
17	0F	13	B0	FF	10	00	00	50
16	0F	C2	9F	FF	10	00	00	50
15	0F	CF	F7	FF	10	00	00	50
14	0F	8E	DE	FF	10	00	00	50
13	0F	8D	B6	FF	10	00	00	50
12	0F	4C	9D	FF	10	00	00	50
11	0F	4B	F5	FF	10	00	00	50
10	0F	0A	DC	FF	10	00	00	50
9	0F	09	B4	FF	10	00	00	50
8	0F	C8	9B	FF	10	00	00	50
7	0F	C7	F3	FF	10	00	00	50
6	0F	86	DA	FF	10	00	00	50
5	0F	85	B2	FF	10	00	00	50
4	0F	44	99	FF	10	00	00	50
3	0F	43	F1	FF	10	00	00	50
2	0F	02	D8	FF	10	00	00	50
1	0F	01	B0	FF	10	00	00	50
0	Base 31…24	Attribute		Base 23…0			Limit 15…0	

Таблиця Д2 - Local Descriptor Table

60	00	00	CC	05	00	0C	0X	01
59	00	00	EC	06	00	10	0X	02
58	00	00	AC	07	00	1C	0X	03
57	00	00	CC	08	00	20	0X	04
56	00	00	EC	09	00	2C	0X	05
55	00	00	AC	0A	00	30	0X	06
54	00	00	CC	0B	00	3C	0X	07
53	00	00	EC	0C	00	40	0X	08
52	00	00	AC	0D	00	4C	0X	09
51	00	00	CC	0E	00	50	0X	10
50	00	00	EC	0F	00	5C	0X	11
49	00	00	AC	10	00	60	0X	12
48	00	00	CC	11	00	6C	0X	13
47	00	00	EC	12	00	70	0X	14
46	00	00	AC	13	00	7C	0X	15
45	00	00	CC	14	00	80	0X	16
44	00	00	EC	15	00	8C	0X	17
43	00	00	AC	16	00	90	0X	18
42	00	00	CC	17	00	9C	0X	19
41	00	00	EC	18	00	A0	0X	20
40	00	00	AC	19	00	AC	0X	21
39	00	00	CC	1A	00	B0	0X	22
38	00	00	EC	1B	00	BC	0X	23
37	00	00	AC	1C	00	C0	0X	24
36	00	00	CC	1D	00	CC	0X	25
35	00	00	EC	1E	00	D0	0X	26
34	00	00	AC	1F	00	DC	0X	27
33	00	00	CC	00	00	E0	0X	28
32	00	00	EC	01	00	EC	0X	29
31	00	00	AC	02	00	F0	0X	30
	Offset 31…16		Attribute		Selector		Offset 15…0	

Таблиця Д2 - Local Descriptor Table (закінчення)

30	0F	01	B0	FF	20	00	00	30
29	0F	02	D8	FF	20	00	00	30
28	0F	43	F1	FF	20	00	00	30
27	0F	44	99	FF	20	00	00	30
26	0F	85	B2	FF	20	00	00	30
25	0F	86	DA	FF	20	00	00	30
24	0F	C7	F3	FF	20	00	00	30
23	0F	C8	9B	FF	20	00	00	30
22	0F	09	B4	FF	20	00	00	30
21	0F	0A	DC	FF	20	00	00	30
20	0F	4B	F5	FF	20	00	00	30
19	0F	4C	9D	FF	20	00	00	30
18	0F	8D	B6	FF	20	00	00	30
17	0F	8E	DE	FF	20	00	00	30
16	0F	CF	F7	FF	20	00	00	30
15	0F	C2	9F	FF	20	00	00	30
14	0F	13	B0	FF	20	00	00	30
13	0F	14	D8	FF	20	00	00	30
12	0F	55	F1	FF	20	00	00	30
11	0F	56	99	FF	20	00	00	30
10	0F	97	B2	FF	20	00	00	30
9	0F	98	DA	FF	20	00	00	30
8	0F	D9	F3	FF	20	00	00	30
7	0F	DA	9B	FF	20	00	00	30
6	0F	1B	B4	FF	20	00	00	30
5	0F	1C	DC	FF	20	00	00	30
4	0F	5D	F5	FF	20	00	00	30
3	0F	5E	9D	FF	20	00	00	30
2	0F	9F	B6	FF	20	00	00	30
1	0F	91	DE	FF	20	00	00	30
0	Base 31…24	Attribute		Base 23…0			Limit 15…0	

Таблиця Д3 - Interrupt Descriptor Table

30	00	00	8E	00	00	0C	0A	01
29	00	00	8F	00	00	10	0B	02
28	00	00	8E	00	00	1C	0C	03
27	00	00	8F	00	00	20	0A	04
26	00	00	8E	00	00	2C	0B	05
25	00	00	8F	00	00	30	0C	06
24	00	00	8E	00	00	3C	0A	07
23	00	00	8F	00	00	40	0B	08
22	00	00	8E	00	00	4C	0C	09
21	00	00	8F	00	00	50	0A	10
20	00	00	8E	00	00	5C	0B	11
19	00	00	8F	00	00	60	0C	12
18	00	00	8E	00	00	6C	0A	13
17	00	00	8F	00	00	70	0B	14
16	00	00	8E	00	00	7C	0C	15
15	00	00	8F	00	00	80	0A	16
14	00	00	8E	00	00	8C	0B	17
13	00	00	8F	00	00	90	0C	18
12	00	00	8E	00	00	9C	0A	19
11	00	00	8F	00	00	A0	0B	20
10	00	00	8E	00	00	AC	0C	21
9	00	00	8F	00	00	B0	0A	22
8	00	00	8E	00	00	BC	0B	23
7	00	00	8F	00	00	C0	0C	24
6	00	00	8E	00	00	CC	0A	25
5	00	00	8F	00	00	D0	0B	26
4	00	00	8E	00	00	DC	0C	27
3	00	00	8F	00	00	E0	0A	28
2	00	00	8E	00	00	EC	0B	29
1	00	00	8F	00	00	F0	0C	30
0	Offset 31…16		Attribute		Selector		Offset 15…0	