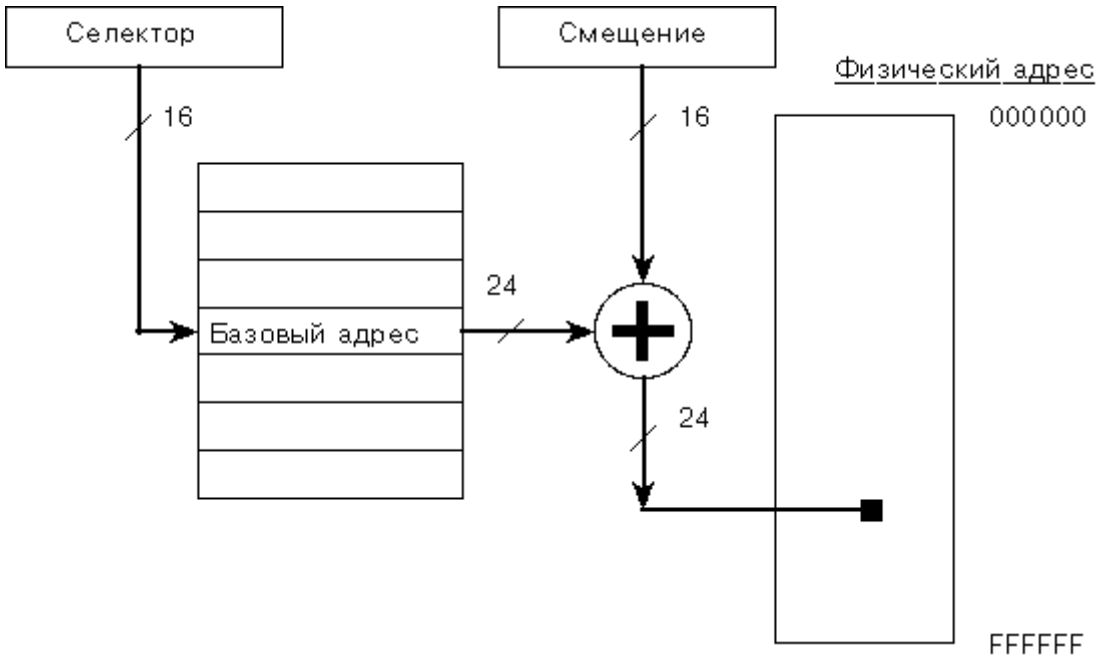
Основная трудность, с которой сталкивается программист, изучающий защищённый режим - это сложная схема преобразования адресов. Поэтому мы уделим много внимания тому, как в этом режиме происходит адресация памяти. Для облегчения восприятия мы вначале опустим некоторые детали, отложив на время их полное описание.

Преобразование адресов в защищённом режиме

В защищённом режиме, также как и в реальном, существуют понятия логического и физического адреса. Логический адрес в защищённом режиме (иногда используется термин "виртуальный адрес") состоит из двух 16-разрядных компонент - селектора и смещения. Селектор записывается в те же сегментные регистры, что и сегментный адрес в реальном режиме. Однако преобразование логического адреса в физический выполняется не простым сложением со сдвигом, а при помощи специальных таблиц преобразования адресов.

В первом приближении можно считать, что для процессора i80286 селектор является индексом в таблице, содержащей базовые 24-разрядные физические адреса сегментов. В процессе преобразования логического адреса в физический процессор прибавляет к базовому 24-разрядному адресу 16-разрядное смещение, т.е. вторую компоненту логического адреса.

На рисунке показана сильно упрощённая схема преобразования логического адреса в физический.

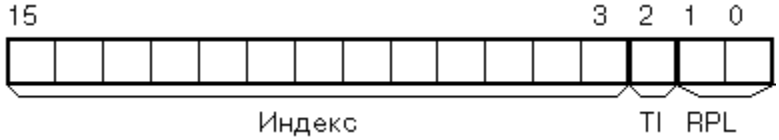


Такая схема формирования физического адреса позволяет непосредственно адресовать 16 мегабайт памяти с помощью 16-разрядных компонент логического адреса.

Заметьте, что селектор - это не сегментный адрес. Это индекс, с помощью которого процессор извлекает из специальной таблицы 24-разрядный базовый адрес сегмента. В реальном режиме мы имеем дело с сегментным адресом и смещением, а в защищённом - с селектором и смещением.

На самом деле не все 16 бит селектора используются для индексации по таблице базовых адресов. В качестве индекса выступают старшие 13 бит. Два младших бита (бит 0 и бит 1) используются системой защиты памяти, о чём мы подробно поговорим в следующем разделе. Бит 2 позволяет выбирать для преобразования адреса один из двух типов таблиц преобразования адресов.

Полный формат селектора показан ниже



На этом рисунке два младших бита обозначены как RPL (Requested Privilege Level). Это поле является запрошенным программой уровнем привилегий и его мы будем обсуждать позже. Поле TI (Table Indicator) состоит из одного бита. Если этот бит равен нулю, для преобразования адреса используется так называемая глобальная таблица дескрипторов GDT (Global Descriptor Table), в противном случае - локальная таблица дескрипторов LDT (Local Descriptor Table).

Таблица дескрипторов - это просто таблица преобразования адресов, содержащая базовые 24-разрядные физические адреса сегментов и некоторую другую информацию. То есть каждый элемент таблицы дескрипторов (дескриптор) содержит 24-разрядный базовый адрес сегмента и другую информацию, описывающую сегмент.

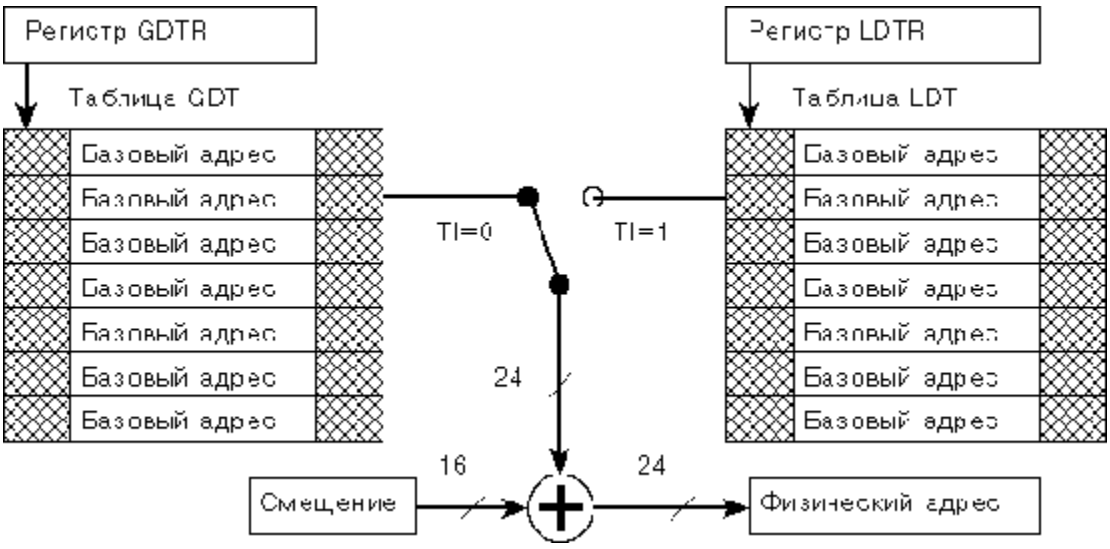
Таблица GDT - единственная в системе. Обычно в ней находятся описания сегментов операционной системы. Таблиц LDT может быть много. Эти таблицы содержат описания сегментов программ, работающих под управлением операционной системы, т.е. отдельных задач. В каждый данный момент времени процессор может использовать только одну таблицу LDT.

Процессор имеет два регистра, предназначенных для адресации используемых в настоящий момент таблиц GDT и LDT. Регистр GTDR описывает расположение и размер таблицы GDT, а регистр LDTR содержит ссылку на использующуюся в настоящее время таблицу LDT.

На рис. 6 показана уточнённая схема преобразования адресов в защищённом режиме. Из рисунка видно, что регистры процессора GDTR и LDTR определяют расположение в памяти таблиц GDT и LDT соответственно. Таблицы GDT и LDT содержат дескрипторы, описывающие сегменты памяти. В этих дескрипторах, помимо другой информации (заштрихованная область) содержится 24-разрядный базовый адрес сегмента.

Старшие 13 битов селектора (индекс) выбирают элемент из таблицы GDT или LDT в зависимости от состояния бита TI селектора.

Извлечённый из таблицы дескрипторов базовый адрес сегмента складывается процессором для получения 24-разрядного физического адреса.



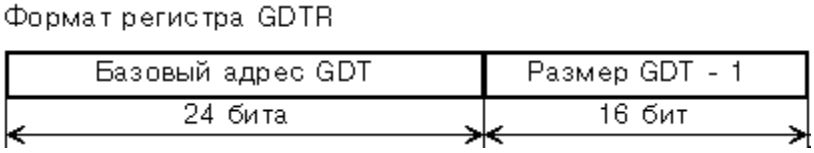
Селектор 0000h адресует самый первый дескриптор в глобальной таблице дескрипторов GDT. Поле RPL для этого дескриптора равно 0, поле TI также равно 0. Селектор 0008h указывает на второй элемент таблицы GDT, а селектор 0014h указывает на третий дескриптор в локальной таблице дескрипторов LDT, т.к. поле TI в нём равно 1.

Детальное описание схемы преобразования адресов

Теперь перейдём к более строгому описанию схемы преобразования адресов в защищённом режиме.

Регистр GDTR, указывающий расположение в физической памяти и размер глобальной таблицы дескрипторов GDT является ключевым в схеме адресации защищённого режима.

Формат регистра GDTR процессора i80286 приведён ниже



Из рисунка видно, что регистр GDTR имеет длину 5 байт. Старшие 3 байта содержат 24-разрядный физический адрес таблицы GDT, младшие два байта - длину таблицы GDT, уменьшенную на 1.

Длина GDT, уменьшенная на единицу, называется пределом таблицы GDT (GDT limit). Она используется для проверки правильности задаваемых программой селекторов. Поле индекса селектора должно содержать ссылки только на существующие элементы таблицы GDT, в противном случае произойдет прерывание. Зная размер GDT, процессор блокирует использование селекторов со значениями поля индекса, выходящее за рамки разрешённых для таблицы GDT. Аналогичный механизм используется и для проверки селекторов, ссылающихся на LDT.

Перед переходом в защищённый режим программа должна создать в оперативной памяти таблицу GDT и загрузить регистр GDTR при помощи специальной команды LGDT (синтаксис транслятора Turbo Assembler, режим IDEAL):

```
lgdt    [QWORD gdt_ptr]
```

Перед выдачей команды LGDT gdt_ptr необходимо подготовить область памяти с адресом gdt_ptr, записав в неё физический адрес таблицы GDT и её размер, уменьшенный на 1 (предел):

```
gdt_ptr dw      GDT_LIMIT       ; предел таблицы GDT
base_lo dw      ?               ; младшее слово базового адреса GDT
base_hi dw      ?               ; старшее слово базового адреса GDT
```

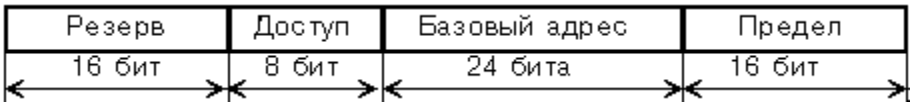
Заметьте, что несмотря на то что размер регистра GDTR составляет 5 байт, в качестве операнда для команды LGDT используется адрес области памяти размером 6 байт. Здесь нет никакой ошибки, процессор i80286 использует только 5 байт из этой области, так как физический адрес содержит 24 разряда. Процессоры i80386 и i80486 в 32-разрядном режиме используют все 6 байтов, загружая в 6-байтный регистр GDTR 32-битовый физический адрес таблицы GDT и её 16-битовый предел.

Однако мы ещё не знаем точную структуру таблицы GDT. Сначала мы познакомим вас со структурой таблицы GDT (и соответственно, с идентичной ей структурой таблицы LDT), а затем на примере фрагмента программы покажем, как создать таблицу GDT и загрузить регистр GDTR.

Как мы уже говорили, таблицы GDT и LDT представляют собой массивы дескрипторов - описателей сегментов. Кроме дескрипторов, описывающих сегменты памяти, таблица GDT может содержать специальные типы дескрипторов - вентили вызова (call gate), задач (task gate) и ловушек (trap gate).

Вентили определяют точки входа в соответствующие процедуры. Например, вентиль вызова описывает адрес подпрограммы, вызываемой, например, по команде CALL. При вызове подпрограммы через вентиль в качестве операнда для команды CALL используется селектор, адресующий соответствующий дескриптор в таблице GDT (или в таблице LDT).

На рисунке приведён формат дескриптора сегмента для процессора i80286:



Длина дескриптора составляет 8 байт. Он состоит из следующих полей:

- поле базового адреса длиной 24 бита содержит физический адрес сегмента, описываемого данным дескриптором;
- поле предела содержит размер сегмента в байтах, уменьшенный на единицу;
- поле доступа описывает тип сегмента (сегмент кода, сегмент данных и др.);
- зарезервированное поле длиной 16 бит для процессора i80286 должно содержать нули, это поле используется процессорами i80386 и i80486 (там, в частности, хранится старший байт 32-разрядного базового адреса сегмента).

В реальном режиме начало сегмента в памяти определяется сегментным адресом, длина сегмента составляет 64 килобайта. В защищённом режиме можно задавать сегменты меньшего размера (процессоры i80386 и i80486 допускают также существование сегментов с размером, значительно превышающим 64 килобайта). При этом процессор автоматически отслеживает попытки программы обратиться за пределы сегмента, заданные в дескрипторе. При обнаружении такой попытки выполнение программы прерывается.

Ограничение размера сегментов и контроль за попытками адресации памяти вне пределов сегментов сильно повышает надёжность системы. Программа не может разрушить чужие сегменты, в частности, сегменты операционной системы. Подробнее о защите мы расскажем в следующем разделе.

Поле доступа, занимающее в дескрипторе один байт (байт доступа) служит для классификации дескрипторов. Ниже приведены форматы поля доступа для трёх типов дескрипторов - дескрипторов сегментов кода, сегментов данных и системных.

Поле доступа дескриптора сегментов кода содержит битовое поле R, называемое битом разрешения чтения сегмента. Если этот бит установлен в 1, программа может считывать содержимое сегмента кода. В противном случае процессор может только выполнять этот код.

Программа не может модифицировать сегмент кода. Это означает невозможность создания самомодифицирующихся программ для защищённого режима (распространённая, но порочная практика среди программистов, создающих программы для MS-DOS). Впрочем, есть обходной путь. Для сегмента кода можно создать ещё один, алиасный дескриптор, в котором этот сегмент отмечен как сегмент данных. Если для этого сегмента будет разрешена запись, ничто, кроме здравого смысла, не помешает вам модифицировать код программы во время её выполнения.

Бит C называется битом подчинения, он будет рассмотрен в следующем разделе.

Биты P и A предназначены для организации виртуальной памяти. Их назначение будет описано в разделе, посвящённом виртуальной памяти. Сейчас отметим, что бит P называется битом присутствия сегмента в памяти. Для тех сегментов, которые находятся в физической памяти (мы будем иметь дело в основном с такими сегментами) этот бит должен быть установлен в 1.

Любая попытка программы обратиться к сегменту памяти, в дескрипторе которого бит P установлен в 0, приведёт к прерыванию.

Бит A называется битом обращения к сегменту и для всех наших программ должен быть установлен в 0.

Поле доступа дескриптора сегмента данных имеет битовые поля W и D. Поле W называется битом разрешения записи в сегмент. Если этот бит установлен в 1, наряду с чтением возможна и запись в данный сегмент. В противном случае при попытке чтения выполнение программы будет прервано.

Поле D задаёт направление расширения сегмента. Обычный сегмент данных расширяется в область старших адресов (расширение вверх). Если же в сегменте расположен стек, расширение происходит в обратном направлении - в область младших адресов (расширение вниз). Для сегментов, в которых организуются стеки, необходимо устанавливать поле D равным 1.

Дескрипторы системных сегментов содержат поле TYPE, определяющее тип системного сегмента. В таблице 1 приведены возможные для этого поля значения.

Таблица 1. Типы системных сегментов.

Поле TYPE	Тип сегмента
0	Запрещённое значение
1	Доступный TSS для процессора i80286
2	Сегмент LDT
3	Занятый TSS для процессора i80286
4	Вентиль вызова для процессора i80286
5	Вентиль задачи для процессоров i80286 и i80386
6	Вентиль прерывания для процессора i80286
7	Вентиль исключения для процессора i80286
8	Запрещённое значение
9	Доступный TSS для процессора i80386
A	Зарезервировано
B	Занятый TSS для процессора i80386
C	Вентиль вызова для процессора i80386
D	Зарезервировано
E	Вентиль прерывания для процессора i80386
F	Вентиль ловушки для процессора i80386