**ПЕРЕДАЧА УПРАВЛЕНИЯ МЕЖДУ УРОВНЯМИ ПРИВИЛЕГИЙ**

**Напомним основные моменты защиты по привилегиям, рассмотренные в предыдущих разделах:**

**- процессор i86 аппаратно поддерживает работу на любом из четырех уровней привилегий;**

**- текущий уровень привилегий CPL определяется выполняемым кодом, который, в свою очередь, задается полем DPL дескриптора сегмента кода;**

**- процессор жестко ограничивает передачи управления в пределах текущего уровня привилегий.**

**Передача управления между уровнями привилегий в первую очередь связана с доступом пользовательских программ (PL3-программ) к обслуживаниям операционной системы, процедуры которых работают на уровне PL1. Об объемах услуг операционной системы свидетельствует, например, тот факт, что операционная система представляет пользователям более 150 разнообразных обслуживаний.**

**Имеются два ответа на поставленный выше вопрос. Первый их них (простой) заключается в применении подчиненных сегментов кода. Второй (сложный) – в использовании специальных дескрипторов, называемых шлюзами вызова.**

**ПОДЧИНЕННЫЕ СЕГМЕНТЫ КОДА.**

**Представим себе, что в системе имеется процедура (т.е. сегмент кода) для преобразования двоичных целых чисел в символьный код ASCII, который используют программы на каждом уровне привилегий. Так как вызовы менее привилегированных процедур процессор запрещает, процедура должна находиться в кольце 0, чтобы ее можно было вызвать из этого кольца. Обычно такая процедура требует достаточных привилегий только для обращения к своему параметру (двоичному числу) и возвращения результата (символьного кода), т.е. таких же привилегий, что и вызывающая программа. Именно для таких ситуаций в процессоре i86 предусмотрены подчиненные (точнее, подчиняемые) или согласованные сегменты кода.**

**Сегмент кода определяется, как подчиненный установкой бита 2 (бит C – Conforming) в байте прав доступа AR дескриптора сегмента – типы сегментов 6 и 7. В таких сегментах может быть разрешена или запрещена операция считывания. Обычные правила защиты по значениям CPL и DPL не действуют, если бит C=1. В этом случае вводятся другие правила.**

**С подчиненными сегментами кода не ассоциируется конкретный уровень привилегий, так как они подчиняются уровню привилегий того вида, который передает им управление с помощью команд CALL или JMP. Другими словами, подчиненные сегменты кода никогда не требуют больше привилегий, чем код, передающий им управление. Если, например, PL3-программа передает управление подчиненному сегменту кода, то он работает с CPL = 3; если же этот сегмент вызывает PL0-программа, то он выполняется с CPL = 0. Из-за такой подвижности уровня привилегий подчиненного сегмента кода в нем не должны содержаться приведенные выше привилегированные команды, например, команды ввода-вывода.**

**Когда управление передается подчиненному сегменту кода, биты поля RPL регистра CS не изменяются на поле DPL дескриптора нового сегмента кода, как это обычно бывает, а сохраняют значения DPL последнего выполнявшегося неподчиненного сегмента кода. Только в этой единственной ситуации биты RPL регистра CS не соответствуют битам DPL дескриптора текущего выполняющегося сегмента кода.**

**Хотя с подчиненным сегментом кода не ассоциируется никакой конкретный уровень привилегий, применительно к нему все же действует одно ограничение. Оно заключается в том, что значение DPL дескриптора подчиненного сегмента кода всегда должно быть меньше или равно текущему значению CPL. Другими словами, передача управления разрешается только во внутренние, более защищенные кольца. Передавать управление сегменту с DPL > CPL (т.е. менее привилегированному, чем текущая программа) никогда не разрешается.**

**Чтобы разобраться в упомянутом ограничении, следует рассмотреть ситуацию, когда процедура в подчиненном сегменте закончилась и необходимо возвращать управление на уровень привилегий вызвавшего кода. Если для перехода к подчиненному сегменту кода применялась команда JMP (или команда CALL), то возвращающая управление команда JMP (или команда CALL) должна передавать управление более привилегированному и, главное, неподчиненному сегменту кода. Такие передачи управление запрещаются, что, собственно, лежит в основе механизма защиты по привилегиям в процессоре i86.**

**ШЛЮЗЫ ВЫЗОВА. Наличие подчиненных сегментов кода обеспечивает некоторую свободу передачи управления с одного уровня привилегий на другой. Хотя подчиненный код фактически не изменяет текущий уровень привилегий, он позволяет программам на различных уровнях привилегий разделять общий код. Обычно им являются разделенные библиотеки, а идея подчиненного кода не требует наличия копий библиотек на каждом уровне привилегий.**

**Для реализации фактического изменения уровней привилегий привлекаются особые системные объекты, называемые шлюзами вызова. (В переводной литературе употребляется менее удачный термин «вентиль вызова».)**

**В дополнение к дескрипторам сегментов кода и данных прикладных программ процессор i86 имеет дескрипторы для системных сегментов и шлюзов. Эти структуры данных привлекаются для управления задачами, а также особыми случаями и прерываниями. В таблице 1 приведены все типы дескрипторов, определенные для системных сегментов и шлюзов. Отметим, что не все дескрипторы определяют сегменты; например, в дескрипторах шлюзов содержатся точки входа процедур.**

**Таблица 1. Типы системных сегментов и шлюзов**

|  |  |
| --- | --- |
| **Тип** | **Описание** |
| **0**  **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15** | **Зарезервирован**  **Доступный TSS процессора 80286**  **Таблица LDT**  **Занятый TSS процессора 80286**  **Шлюз вызова**  **Шлюз задачи**  **Шлюз прерывания процессора 80286**  **Шлюз ловушки процессора 80286**  **Зарезервирован**  **Доступный TSS процессора i486**  **Зарезервирован**  **Занятый TSS процессора i86**  **Шлюз вызова процессора i86**  **Зарезервирован**  **Шлюз прерывания процессора i86**  **Шлюз задачи процессора i86** |

**Как системный объект, шлюз вызова имеет собственный дескриптор:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 48 | | 47 | | | | 16 | 15 | |  | | | 0 | | |
| Селектор | | | Offset(Смещение в сегменте) | | | | | Атрибуты | | | | | | | |
|  | | |  | |  |  | |  | | | | | | | |
| Индекс(13 бит) | | G/L | | RPL(2 бита) |  | Счетчик двойных слов | | P | DPL | | 0 | Тип (4бита) | | |  |
| Рис.3.9. Структура дескриптора шлюза программных сегментов. | | | | | | | | | | | | | |  |  |

**Селектор сегмента**

Все дескрипторы, кроме дескрипторов прерываний (дескрипторы сегментов, задач, шлюзов сегментов или задач), хранятся в оперативной памяти в специальных таблицах GDT(глобальная таблица дескрипторов) иLDT(локальная таблица дескрипторов).

В глобальной таблице дескрипторов хранятся все сегменты операционной системы.

Локальные таблицы дескрипторов формируются на каждую задачу (программу, находящуюся в стадии исполнения, возможно, с разделением по времени). Но в каждый момент времени активны только две таблицы: GDTи одна изLDT. Обращение к таблицам производится по селектору сегмента.

Селектор сегмента определяет дескриптор используемого кодового сегмента, т.е. целевой сегмент. Селектор сегмента содержит следующие поля:

* индекс (13 бит) – используется как адрес в таблице целевого дескриптора (указателя "целевой" строки таблицы дескрипторов),
* бит типа "целевой" таблицы дескрипторов; при нулевом значении G/L **–** выбираетсяGDT, иначеLDT**,**
* поле уровня привилегии запроса (в рассматриваемой конструкции не используется).

Бит G/Lселектора допускает обращение к дескрипторам задач как задачам операционной системы, так и пользовательским. Но это выборочный доступ только к определенным "сервисным процедурам", предоставление которых пользовательским задачам является одной из главных функций ОС. Контроль корректности использования сервисных процедур задачами пользователей осуществляется механизмом проверок обращений по уровням привилегий.

**Смещение в сегменте (offset)**

Смещение в сегментев программных сегментах определяет точку входа в сегмент. (По селектору дескриптора шлюза определяется целевой кодовый сегмент, а по смещению в сегменте – адрес первой команды).

**Атрибуты сегмента**

Атрибуты сегмента определяют тип сегмента, права использования, доступность программной процедуры по уровням привилегий, а также задают размер параметров, передаваемых в процедуру через стек.

**Процедура переключения задач**

Межсегментные переключения задач – это переходы с использованием механизма переключения задач. Возможны две модели переключения задач:

* прямое переключение задач;
* переключение задач с использованием шлюзов (косвенное переключение задач).

Модели переключения задач могут использоваться

* для переключения независимых задач при параллельном их выполнении в режиме разделения времени;
* для вызова процедур:
* для вызова программ обработки прерываний и ловушек;
* возврата из программ обработки прерываний и ловушек.

**Переключение задач при параллельном выполнении программ в режиме разделения времени.**

Переключение задач производится с использованием специальных сегментов состояния задач (TSS). Это системные сегменты, предназначенные для копирования основных программно-доступных регистров процессора, хранящих "контекст" программы. Иногда переключение задач называют сменой контекста. В многозадачном режиме работы в стадии выполнения могут находиться несколько задач. Для каждой из них создается сегмент состояния задачи – TSS. Выполнение этих задач может производиться одним процессором в режиме разделения времени. Основным назначением механизма переключения задач является организация очередных переходов между выполняемыми программами.

Переключение программ может производиться командами JMP и CALL типа FAR (межсегментные переходы), командами вызова прерываний, например, INT n, или командой IRET. Кроме этого, переключения задач могут инициироваться прерываниями и ловушками.

В защищенном режиме команды межсегментных переходов содержат полный указатель CS и IP (адрес команды в сегменте –offset). Указатель CS**,** в свою очередь, может указывать на:

* дескриптор сегмента,
* шлюз дескриптора сегмента,
* дескриптор TSS,
* дескриптор шлюза TSS.

Все зависит от выбора стратегии разработчиками ОС.

В первых двух случаях реализуется процедура передачи управления:

* прямой передачи управления (без изменения уровней привилегий), если селектор указывает на дескриптор сегмента,
* передачи управления с возможностью увеличения уровня привилегий, (по командам, предусматривающим возвращение, – например Call), если селектор указывает на шлюз дескриптора сегмента

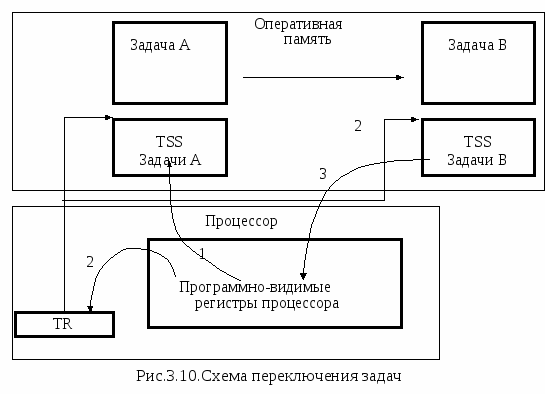
В двух последних случаях реализуется процедура переключения задач:

* прямого переключения задач, если селектор указывает на дескриптор задачи (используется при переключениях задач на одном уровне привилегий),
* косвенного переключения задач, если селектор указывает на шлюз задачи (используется при переключениях с изменениями уровня привилегий)**.**

При косвенном переключении задач производится дополнительное обращение к таблице дескрипторов GDT.. Обращение производится по значению селектора дескриптора целевого кодового сегмента, указанного в шлюзе.

Сегмент состояния задачи (TSS) является сегментом, но системным. Дескриптор TSS по структуре совпадает с дескриптором программных сегментов (см. рис. 3.8.), в поле атрибутов которых отмечено, что они системные.

С  
**Структура дескриптора шлюза TSSсовпадает со структурой дескриптора шлюза программных сегментов (см. рис.3.9). Отличие заключаются в содержимом поля атрибутов и в том, что не используется поле Offset(Смещение в сегменте).**



**Процедура переключения задач состоит из 3-х пересылок   
(см. рис.3.10):**

1. **сохранение содержимого всех программно-доступных регистров процессора в сегменте TSS; базовый адрес сегмента TSS берется из системного регистра TR,**
2. **адрес сегмента TSS новой задачи по селектору из прямой команды перехода или по содержимому поля CS дескриптора шлюза задачи переписывается в системный регистр TR.**
3. **содержимое TSS новой задачи переписывается в регистры процессора, и процессор начинает выполнение программы-цели.**

**Длительность процедуры переключения задач – 200 тактов.**

**С точки зрения взаимодействия программ, переключение программ равнозначно завершению одной программы с сохранением всего контекста программы и запуску следующей программы. При переключении задач запоминается текущее значение указателя команд IP. Поэтому при ее повторном запуске задача запускается не с начальной точки, а продолжает выполнение с прерванной точки.**

**До аппаратной реализации этой процедуры, например в реальном режиме работы, переключение задач (в многозадачных режимах работы) производилось чисто программными способами, что уменьшало производительность процессора.**

**Обработка ловушек**

**Обработка ловушек – это обработка события, непосредственно связанная с выполнением текущей программы. Следовательно, обработчик прерывания должен попадать в контекст выполняемой программы. В общем случае, смена контекста здесь противопоказана. Но в некоторых случаях механизм переключения задач и при обработке ловушек может быть полезен.**

**Прерывания**

**Прерывания служат для обработки некоторого внешнего события, явно не связанного с прерываемой программой. Для этих случаев целесообразно использовать переключение задач. Выход из подпрограмм (обратное переключение) производится командой IRET*.***

**Механизм выполнения команды IRET**

**Механизм выполнения команды IRET определяется способом вызова процедуры обработки прерывания.**

**Если переход на программу обработки был произведен с использованием механизма переключения программ, то и возврат на исходную программу производится переключением задач.**

**Если переключение задач вызывается командами CALL, командой вызова прерываний INTn, или прерыванием, то процессор устанавливает бит вложенности NT в регистре флагов FLAGSи бит занятости B в поле прав доступа дескриптора. Затем записывает селектор дескриптора TSS прерываемой программы в специальное поле "селектора возврата" TSS принимающей программы, и производится процедура переключения задач по схеме 3.10. После этого обе задачи становятся занятыми. Это запрещает применение рекурсивных процедур и реентерабельных программ.**

**Если переключение задач вызывает команда JMP, то селектор возврата, бит вложенности и бит занятости не устанавливаются*.***

**При выполнении команды возврата процессор проверяет бит вложенности (NT) и выбирает механизм перехода. В случае использования переключения задач читается селектор TSS программы возврата.**

**Отсутствие прямого взаимодействия программ при переходах с использованием механизма переключения задач позволяет значительно смягчить требования к доступности программ по условиям корректности переходов с изменением уровня привилегий. Но все проверки, непосредственно не связанные с уровнями привилегий, выполняются в том же объеме.**

**TSS по чтению и по записи программно недоступен. Создание, модификацию и чтение TSS операционные системы могут производить путем отображения в тот же адрес памяти дескриптора сегмента данных.**

***Вопросы для самопроверки:***

1. ***Назначение дескриптора сегментов в защищенном режиме МП IA-32.***
2. ***Назначение дескриптора шлюзов в защищенном режиме МП IA-32.***
3. ***Процедура переключения задач в защищенном режиме МП IA-32.***
4. ***Основные поля дескрипторов сегментов и шлюзов.***
5. ***Использование поля базового адреса сегмента и поля предела в дескрипторе сегмента.***
6. ***Назначение поля атрибутов в дескрипторе сегмента.***
7. ***Возможности использования сегментов CS,SS,ES,CS.***
8. ***Понятие селектора сегмента.***
9. ***Процедура переключения задач.***
10. ***Термины: TSS, шлюзTSS.***
11. ***Этапы процедуры переключения задач.***

**3.3.2. Модель памяти**

**В защищенном 32-разрядном режиме МП Intelсегментная структура памяти сохранилась, но изменились размеры сегментов и максимальный размер математической памяти:**

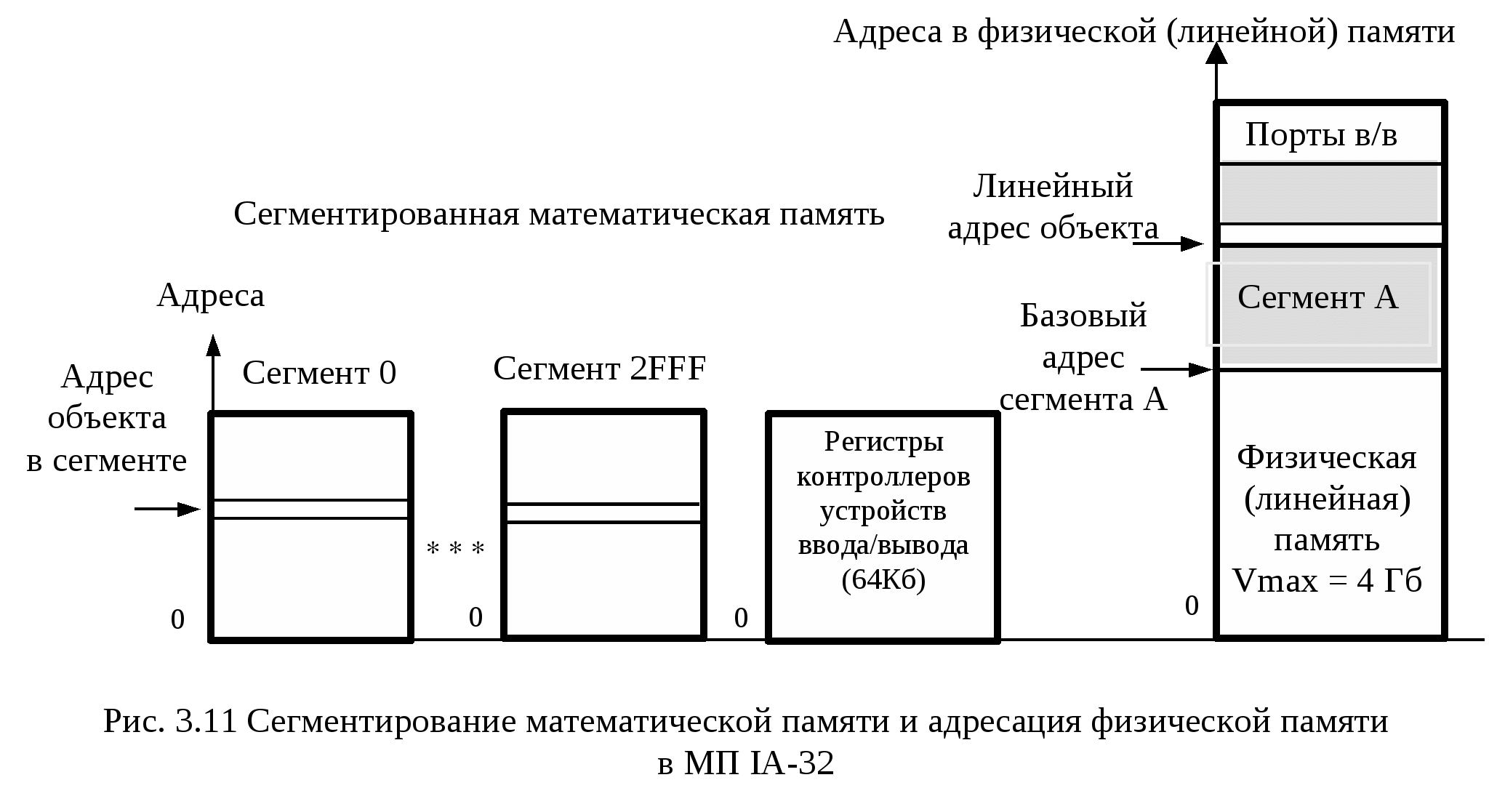
* **максимальный размер сегмента стал равен 232 = 4 294 967 296 байт (4 Гбайта),**
* **максимальное количество сегментов стало равно 214= 16384 (16 Кбайт),**
* **адресное пространство математической памяти стало равно  
  246= 70 368 744 177 664 = (64 Тбайт).**

**Максимальный объем физической линейной памяти определяется разрядностью шин адреса и разрядностью сегментного регистра. При разрядности шин адреса 32 бита возможность наращивания емкости оперативной памяти ограничивается величиной 4 Гбайта, т.е. адресным пространством 1 сегмента.**

**При добавлении разрядных шин адреса ограничение определяется разрядностью сегментного регистра (32 бита). При использовании максимального значения сегмента в 4 Гбайта и максимального значения начального адреса сегмента в линейной памяти возможность наращивания емкости оперативной памяти ограничивается величиной 8 Гбайт (2 сегмента по 4 Гбайт).**

**Биты в памяти и в регистрах также нумеруются с младших разрядов, а многобайтные операнды – по номеру младшего байта. Размещение программных объектов без соблюдения правил целочисленных границ не рекомендуется.**

**Схема сегментированной математической памяти и адресация физической памяти в МП IA-32 представлена на рис. 3.11.**



***Вопросы для самопроверки:***

1. *Максимальный размер сегмента в МП IA-32.*
2. *Максимальное количество сегментов в МП IA-32.*
3. *Чем определяется максимальное количество сегментов в МП IA-32.*
4. *Максимальный размер математической памяти сегментов в МП IA-32.*
5. *Максимальный адрес обращения к оперативной памяти без использования трансляции страниц.*

**Как видно из формата, 48 бит в дескрипторе шлюза вызова определяют полный указатель селектор:смещение точки входа той процедуры (назначения), которой шлюз передаёт управление. Дескриптор шлюза вызова действует как посредник или своеобразный «интерфейсный слой» между сегментами кода, находящимися на различных уровнях привилегий. Шлюзы вызова идентифицируют разрешенные точки входа в более привилегированном коде, которым может быть передано управление, и являются единственным средством смены уровней привилегий. Т.е. шлюз вызова – системный объект, который адресует процедуру в сегменте кода. Шлюз имеет уровень привилегий, отличный от уровня привилегий вызываемого сегмента кода.**

**В отличии от дескрипторов сегментов кода, данных или стека, а также дескрипторов таблиц LDT , дескрипторы шлюзов вызова не определяют никакого адресного пространства, поэтому в них нет полей базы и предела. По существу дескрипторы шлюзов вызова не являются дескрипторами, но их удобно размещать в дескрипторных таблицах. Селекторы, которые выбирают дескрипторы шлюзов вызова можно загружать только в сегментный регистр CS для передачи управления сегменту кода на другом уровне привилегий. Адресовать шлюз вызова разрешается только в команде межсегментного вызова FAR CALL, а команды безусловного перехода JMP запрещены.**