Министерство образования и науки Российской Федерации
Нижнекамский химико-технологический институт (филиал)
федерального государственного бюджетного образовательного учреждения
высшего образования
«Казанский национальный исследовательский технологический университет»
(ФГБОУ ВО «КНИТУ»)

А.В. Долганов

ЭВМ и периферийные устройства

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к лабораторной работе № 1

Нижнекамск 2016

Лабораторная работа № 1

Встроенные средства защиты информации в микропроцессорах Intel

1. ЦЕЛЬ РАБОТЫ

Целью работы является изучение концепции встроенного механизма защиты информации в микропроцессорах Intel.

2. ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.1. НАЗНАЧЕНИЕ, КОНЦЕПЦИИ И КОМПОНЕНТЫ ЗАЩИЩЕННОГО РЕЖИМА.

2.1.1. Назначение и основные концепции.

Защищенный режим работы в МП Intel предусматривает аппаратную поддержку различных вариантов защиты информации от помех. Предусмотренные возможности и глубина защиты определяются используемыми операционными системами.

Необходимость в защите информации появилась при переходе на многопрограммные режимы работы. Было ясно, что ЭВМ, допускающие многопрограммные режимы работы, должны обеспечивать следующие требования:

1. независимость подготовки пользовательских программ и их защита от взаимных помех;
2. защиту программ операционной системы от помех при сбоях в пользовательских программах;
3. защиту программ операционной системы верхнего уровня от помех при сбоях в программах операционной системы нижнего уровня;
4. защиту программ от отрицательных последствий при программных сбоях;
5. защиту целостности системы.

Независимость подготовки пользовательских программ и их защита от взаимных помех. Это требование означает, что:

- при разработке своей программы пользователь не должен учитывать возможное взаимное расположение программ в оперативной памяти и может рассчитывать на любое место в памяти;
- предусматривается локализация возможных негативных последствий программных ошибок в пределах адресных пространств соответствующих программ.

Указанные проблемы в МП Intel в защищенном режиме решаются механизмом управления виртуальной памяти.

Виртуальная память предусматривает гибкое программное распределение ресурсов памяти, динамическую переадресацию и разделение адресных пространств совместно выполняемых программ.

Виртуальная память допускает многопрограммное выполнение прикладных программ, но при этом программы изолируются друг от друга таким образом, что ошибки в одной из них не влияют на корректное выполнение других программ. Когда программа осуществляет "некорректное" обращение к памяти, механизм виртуальной памяти блокирует обращение и сообщает о попытке нарушения защиты.

В МП Intel реализованы два уровня виртуальной памяти - верхний и нижний. Верхний уровень реализован механизмом трансляции сегментов, нижний уровень - механизмом трансляции страниц.

На уровне трансляции сегментов разделение адресных пространств пользовательских программ осуществляется при помощи локальных таблиц дескрипторов сегментов.

Доступ к сегментам осуществляется только через таблицы дескрипторов. Загрузка сегмента в память (оперативную или внешнюю) производится в любую её свободную область. При этом в дескрипторе сегмента фиксируется базовый (начальный) адрес сегмента.

Таким образом, таблицы дескрипторов определяют не только общий объем адресного пространства программы, но и конкретное размещение сегментов программы в физической памяти. Это является основой разделения адресных пространств пользовательских программ.

В каждой пользовательской программы есть своя локальная таблица дескрипторов, недоступная программам других пользователей. Эта таблица определяет доступ к физическим адресам соответствующих сегментов. Следовательно, пользовательские программы могут "видеть" только свое адресное пространство.

Многопользовательские режимы должны не только разделять программы пользователей. Иногда требуется совместная работа нескольких пользовательских программ. Для этого требуются не локальная, а разделяемая память.

Разделяемая память легко организуется при помещении дескрипторов соответствующих сегментов в общую глобальную таблицу дескрипторов или дублированием дескрипторов разделяемых сегментов в локальных таблицах дескрипторов.

На уровне трансляции страниц разделение адресных пространств пользователей производится при разделении таблиц страниц. Каждая программа использует свою таблицу страниц, которая определяет доступ к соответствующим (своим) страницам.

Защита программ операционной системы от помех при сбоях в пользовательских программах.

Выполнение этого требования является более сложной задачей. Дело в том, что операционная система служит не только для организации определенного режима работы и обеспечения "дружелюбного" экранного интерфейса. Одной из важнейших функций операционной системы является предоставление сервисных процедур пользовательским программам. По этой причине глобальная таблица дескрипторов, определяющая сегменты операционной системы,

должна быть доступна пользовательским программам, но не во вред защите программ операционной системы.

Для разрешения этой проблемы используется механизм зашиты по режимам работы (уровням привилегий).

Практически во всех компьютерах для целей защиты предусматриваются как минимум два режима работы: системный режим, называемый также режимом супервизора (Supervisor), и пользовательский режим (User). Основное различие между ними состоит в том, что программам, работающим в режиме супервизора, доступны все ресурсы системы, у программ, работающих в пользовательском режиме, возможности доступа к ресурсам системы ограничены. В современных МП такой механизм защиты называют защитой по уровням привилегий (PL - Privilege Level).

Этот механизм открывает пользовательским программам доступ только к определенным программам операционной системы и обеспечивает достаточно корректное использование этих программ.

Защита программ верхнего уровня операционной системы от помех при сбоях в программах нижнего уровня операционной системы. Такая защита программ необходима, потому что обычно операционные системы имеют иерархическую структуру.

Ошибки на разных уровнях этой структуры имеют различные по тяжести последствия в виде возможных нарушений в работе системы. Естественно, что "жесткость" защиты от помех на разных уровнях иерархии программ операционной системы должна быть различной. По этой причине увеличивают количество уровней привилегий.

В процессоре Intel имеется аппаратная поддержка четырех уровней привилегий. Однако в страничном преобразовании адреса применяется упомянутый простой двухуровневый механизм защиты: пользователь/супервизор.

Защита программ от отрицательных последствий при программных сбоях. Наиболее опасными здесь являются ошибки в использовании адресов данных или переходов. Нередко эти сбои приводят к фатальным последствиям: порче массивов данных и кодов.

Механизм защиты программ от отрицательных последствий при программных сбоях опирается на описание различных системных объектов с помощью дескрипторов.

Дескриптор является "поисковым" образом объекта, определяющим функциональные свойства объекта с учетом определенных требований информационной безопасности. Дескриптор определяет не только место расположения объекта в памяти, но и его тип, степень защиты и права доступа.

На основе информации дескриптора механизм защиты осуществляет блокировку обращения к "неверному" пространству памяти и сообщает о возникновении этой ситуации.

Защита целостности функционирования вычислительной системы. Вычислительные системы проектируются под конкретные режимы работы, определяемые операционной системой. Четкое выполнение административных функций по поддержке заданных режимов входит в понятие целостности вы-

числительной системы. Обеспечение целостности функционирования вычислительной системы реализуется механизмами разделения адресного пространства и защитой по уровням привилегий.

При изучении механизма защиты следует учесть, что:

- защита производится на логическом уровне

Все проверки (например, адресов обращения) осуществляются в процессоре сравнением вычисленных и допустимых эталонных величин. При положительных результатах сравнения адреса посылаются в память. При пересылке адресов и данных возможно их искажение из-за аппаратных сбоев. Механизм защиты перед такими сбоями бессилен. Для борьбы с ними в вычислительных системах используется другой слой защиты - встроенный непрерывной контроль безошибочной работы аппаратуры.

- рассматриваемый механизм является только аппаратной поддержкой защиты. Глубина реализованной защиты определяется операционной системой.

Например, механизм управления виртуальной памятью способен надежно разделять адресные пространства программ, но при условии, что программа-загрузчик операционной системы будет загружать соответствующие сегменты в оперативную память без их перекрытий.

Операционная система должна выполнять противоречивые требования. С одной стороны она должна предоставлять пользователям максимум сервисных услуг, включая услуги по использованию системных ресурсов. С другой стороны для обеспечения целостности системы она должна максимально ограничивать свободу управления ресурсами со стороны пользователей и отделять системные программы от пользовательских.

Как выход из положения, операционные системы используют принцип предоставления услуг пользователям через программы-посредники, контролирующие корректность управления ресурсами. При этом предполагается, что обычные сбои пользовательских программ известны и либо будут отловлены, либо не будут приводить к тяжелым последствиям. Что касается преднамеренного нарушения целостности системы, то здесь остается только надеяться на мастерство программистов - разработчиков операционных систем.

Вопросы для самопроверки:

1. *Что входит в понятие защиты программ от взаимных помех.*
2. *Каким механизмом решается проблема защиты программ от взаимных помех на одном уровне привилегий.*
3. *Как реализуется разделение адресных пространств выполняемых программ.*
4. *В чем заключается основная проблема организации защиты программ операционной системы.*
5. *Заменяет ли работа механизма защиты традиционные средства контроля безошибочной работы ЭВМ.*
6. *Для каких целей в механизме защиты МП корпорации Intel увеличили количество уровней привилегий.*

7. Какую основную функцию в механизме защиты выполняют схемы управления виртуальной памятью.

8. Какую функцию в механизме защиты выполняет процедура трансляция сегментов.

9. Какую функцию в механизме защиты выполняет процедура трансляция страниц.

2.1.2. Компоненты защищенного режима.

Механизм защиты процессора Intel можно подразделить на две части: схемы управления памятью и защита по привилегиям (рис. 1). Схемы управления памятью включают механизм управления виртуальной памятью и контроль обращений к объектам памяти.

Схемы управления памятью обнаруживают большинство программных ошибок, например, формирование неверных адресов, нахождение индекса за пределами массива и т.п.

Защита по привилегиям фиксирует более тонкие ошибки и попытки нарушить целостность функционирования системы.

Рис. 1. Механизм защиты МП Intel.

Схемы управления памятью реализуют разделение адресных пространств различных объектов (программ, данных, таблиц и т.д.), расположенных в оперативной памяти и осуществляют проверку корректности обращения к этим объектам.

Разделение адресных пространств объектов в памяти является одной из основных штатных функций механизма трансляции сегментов и страниц. Дополнительной надстройкой для механизма защиты являются проверки, выпол-

няемые при обращении к памяти, такие как проверка выхода за пределы границ объекта, проверка типа объекта и прав доступа к объекту.

Информационной основой механизма защиты является дескриптор. Дескриптор содержит ссылку на расположение объекта в памяти и описание его свойств (мандатная часть). В поле прав доступа мандатной части содержится основная информация, используемая механизмом защиты. Объектом для дескриптора может служить сегмент (данных, кода, системный и т.д.) или точка входа в сегмент программного кода (шлюз).

Проверки механизма защиты по привилегиям производятся при каждой активизации сегмента. Это или выборка дескриптора сегмента из таблицы дескрипторов при смене сегмента данных, или выборка дескриптора кодового сегмента или шлюза при выполнении межсегментных программных переходов. Проверяется корректность обращений к данным и межсегментных передач управления.

Кроме этого, проверяется корректность использования команд. Корректность использования команд проверяется непрерывно.

Как элемент защиты используются локальные аппаратные стеки для каждого уровня привилегий. При передачах управления с изменением уровня привилегий применяются процедуры переключения стеков.

Микропроцессоры Intel x86 имеют два режима работы: реальный и защищенный. Трансляция страниц, использование дескрипторов и защита по привилегиям производятся только в защищенном режиме.

Микропроцессоры Intel x86 имеют два режима работы: реальный и защищенный. Трансляция страниц, использование дескрипторов и защита по привилегиям производятся только в защищенном режиме.

Считается, что механизм защиты работает параллельно с выполнением программы и не ухудшает производительность процессора.

Это справедливо, если не учитывать некоторые усложнения процедур передач управления, введенные с целью защиты, например, необходимость переключения стеков при изменении текущего уровня привилегий. Но этими потерями можно пренебречь, учитывая низкий процент реализации этих событий.

С другой стороны, защита информации часто является необходимым компонентом вычислений. Многие языки программирования, например Ада, предусматривают простейшие средства контроля. Но при отсутствии аппаратной поддержки, защита организуется программным путем, что значительно увеличивает временные потери. Например, при сложении элементов строк матрицы C(i,j):=A(i,j)+B(j,i) компилятор, при включении опции необязательного контроля выхода адресов за пределы массивов (SUBSCRIPTRANGE), генерирует командный код с избыточностью до 340%.

Вопросы для самопроверки:

1. *Какие основные логические компоненты механизма защиты содержат схемы управления памятью.*
2. *Какие основные проверки при обращении к памяти вы знаете.*

3. Какие механизмы виртуальной памяти реализуют разделение адресных пространств выполняемых программ.

4. Какие компоненты механизма защиты вы знаете.

5. *Какую роль в механизме защиты играют дескрипторы.*

2.2. ИНФОРМАЦИОННАЯ ОСНОВА РАБОТЫ МЕХАНИЗМА ЗАЩИТЫ.

2.2.1. Контрольные поля дескрипторов.

В МП Intel x86 дескрипторы используются как в качестве средств структурирования, так и для защиты программ и данных. В данном разделе дескрипторы рассматриваются только в качестве информационной основы работы механизма защиты.

Описание объектов через дескрипторы используется в различных вычислительных системах. Из пионеров использования дескрипторов можно отметить корпорации Burroughs (Система B5000 - 1961 г.) и Intel (Система iAPX 432 - 1981 г.). Дескрипторы объектов являются средством проектирования структур данных и программ в виде ориентированных графов.

По области использования дескрипторы делятся на системные и пользовательские. По структуре и типу описываемых объектов различают дескрипторы - описатели сегментов и дескрипторы - описатели точек входа в программы (шлюзов).

К описателям сегментов относятся:

- дескрипторы сегментов (кодовых, стековых, данных);
- системные дескрипторы (сегментов состояния задач и различных таблиц, например, таблиц локальных дескрипторов).

К описателям точек входов в программы относятся дескрипторы шлюзов (все являются системными):

- дескрипторы шлюзов вызовов;
- дескрипторы шлюзов прерываний;
- дескрипторы шлюзов ловушек;
- дескрипторы шлюзов задач.

По соображениям совместимости системных программ для МП i80386 и выше с МП i80286 биты некоторых полей дескрипторов оказались расположенными в несмежных разрядах. Но для удобства понимания, структуры дескрипторов сегментов и шлюзов часто представляют в виде логических схем, в которых разряды отдельных полей представляются смежными.

Все дескрипторы содержат по три основных поля. Из них механизм защиты использует следующие поля:

поле атрибутов (в дескрипторах всех типов),

поле предела (в дескрипторах сегментов).

Вопросы для самопроверки:

1. Какие конкретно объекты задаются дескрипторами сегментов.

2. *Какие конкретно объекты задаются системными дескрипторами.*

3. *Какие поля, используемые механизмом защиты, содержат дескрипторы сегментов.*

4. *Какие поля, используемые механизмом защиты, содержат дескрипторы шлюзов.*

2.2.1.1. Контрольные поля дескрипторов сегментов.

Механизм защиты использует два поля дескрипторов сегментов: двадцатиразрядное поле предела и двенадцатиразрядное поле атрибутов (рис. 2).

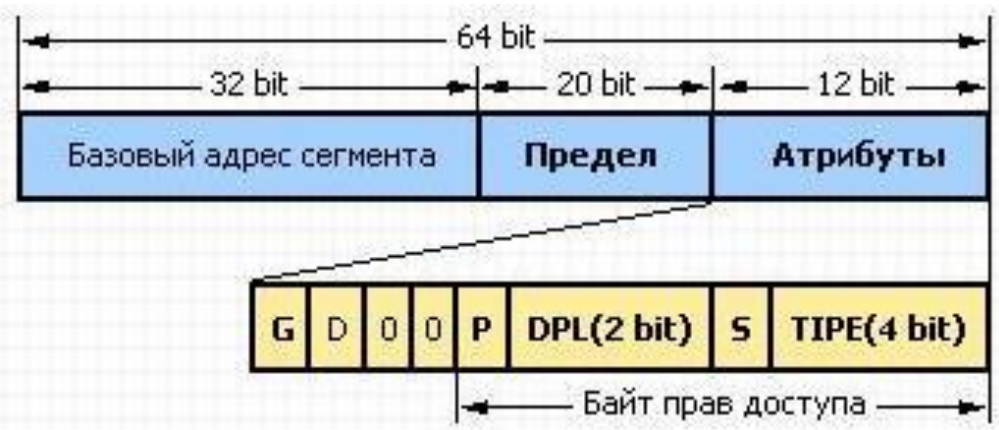


Рис. 2. Структура дескрипторов сегментов.

Поле предела (limit) определяет размер сегмента и используется для контроля выхода за границы сегментов. В поле предела можно задать размер сегмента в байтах - до 1 Мб или в страницах - до 4Гб при размере странице 4Кб.

Поле атрибутов содержит байт прав доступа AR и четыре дополнительных разряда, из которых используются два. Механизм защиты использует только один из них - бит гранулярности G (Granularity). Этот бит определяет меру задания предела сегмента. При G = 0 сегмент задается в байтах, а при G = 1 - в страницах.

Байт прав доступа (рис. 2) содержит следующие поля:

- P (Present) - бит присутствия сегмента в оперативной памяти. Используется механизмом виртуальной памяти для организации подкачки (свопинга) нужного сегмента с диска в оперативную память. Механизмом защиты используется в проверках при загрузке стекового сегмента.
- DPL (Descriptor Privilege Level) - двухразрядное поле, определяющее уровень привилегий сегмента. Механизмом защиты используется в проверках корректности использования сегмента.
- S - тип дескриптора сегмента. При S = 0 - дескриптор определяет сегменты кодов или данных (включая стеки). При S = 1 - дескриптор системный.
- TIPE - четырехразрядное поле, интерпретируемое по-разному в зависимости от типа дескриптора.

В системных дескрипторах (S = 1) разряды поля TIPE кодируют подтипы системных дескрипторов.

В набор системных дескрипторов входят:

- *дескрипторы состояния задач;*
- *дескрипторы локальных таблиц дескрипторов;*
- *дескрипторы шлюзов вызовов;*
- *дескрипторы шлюзов задач;*
- *дескрипторы шлюзов прерываний;*
- *дескрипторы шлюзов ловушек.*

Коды подтипов системных дескрипторов используются механизмом защиты при проверках корректности обращений к программам.

В дескрипторах сегментов данных и кодов (S = 0) поле TIPE (рис. 3) содержит дополнительные признаки сегментов.

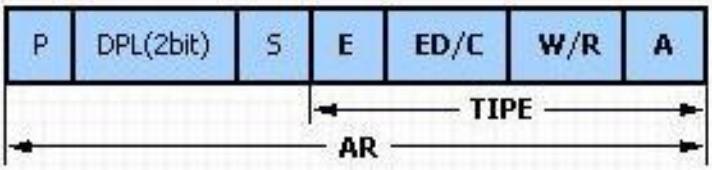


Рис.3. Признаки поля TIPE байта прав доступа AR

В дескрипторах сегментов данных и кодов (S = 0) поле TIPE (рис. 3) содержит следующие признаки:

- *E - исполняемость - уточняет тип сегментов. При E = 0 сегменты определяются как сегменты данных, при E = 1 - как кодовые сегменты. Механизмом защиты используется при проверках корректности обращений к данным и программам;*
- *ED/C - для сегментов данных определяет направление расширения (ED). Механизмом защиты используется при проверках соответствия адреса обращения границам сегмента. Для кодовых сегментов признак ED/C определяет "подчиненность" сегмента (C - conforming). Механизмом защиты используется при проверках корректности межсегментных передач управления,*
- *W/R - определяет права использования. Для кодовых сегментов признак W/R определяет доступность по чтению (R - Read). Модификация кодовых сегментов запрещена безусловно. Для сегментов данных признак W/R определяет доступность по записи (W - Write). Процедура чтения сегментов данных разрешена безусловно. Механизмом защиты используется при проверках корректности использования сегмента;*
- *A - бит обращения к сегменту. Устанавливается в 1 при загрузке селектора или после исполнения команды проверки селектора. Непосредственно механизмом защиты не используется.*

Вопросы для самопроверки:

1. Как задается и как используется механизмом защиты поле длины сегмента.

2. *Как используется бит присутствия сегмента в оперативной памяти.*
3. *Где задается, что означает и как используется поле DPL.*
4. *Какие дескрипторы относятся к системным.*
5. *Что определяет бит E (исполняемость).*
6. *Какие права доступа определяет бит доступности.*

Контрольные поля дескриптора шлюза.

Структура дескрипторов шлюзов представлена на рис. 4. Дескриптор шлюза содержит:

- селектор;
- смещение в сегменте;
- поле атрибутов.

В свою очередь селектор содержит:

- индекс;
- указатель используемой таблицы дескрипторов G/L;
- уровень привилегий источника запроса RPL (Requested Privilege Level).

Селектор дескриптора шлюза используется в качестве указателя дескриптора целевого кодового сегмента в процедурах передач управления. Механизм защиты использует индекс селектора при проверках на возможность выхода обращения за пределы таблицы дескриптора. Остальные поля селектора, включая поле RPL, механизмом защиты не используются.

В поле атрибутов механизм защиты использует только байт прав доступа. Использование этого поля аналогично его использованию в дескрипторе сегмента.

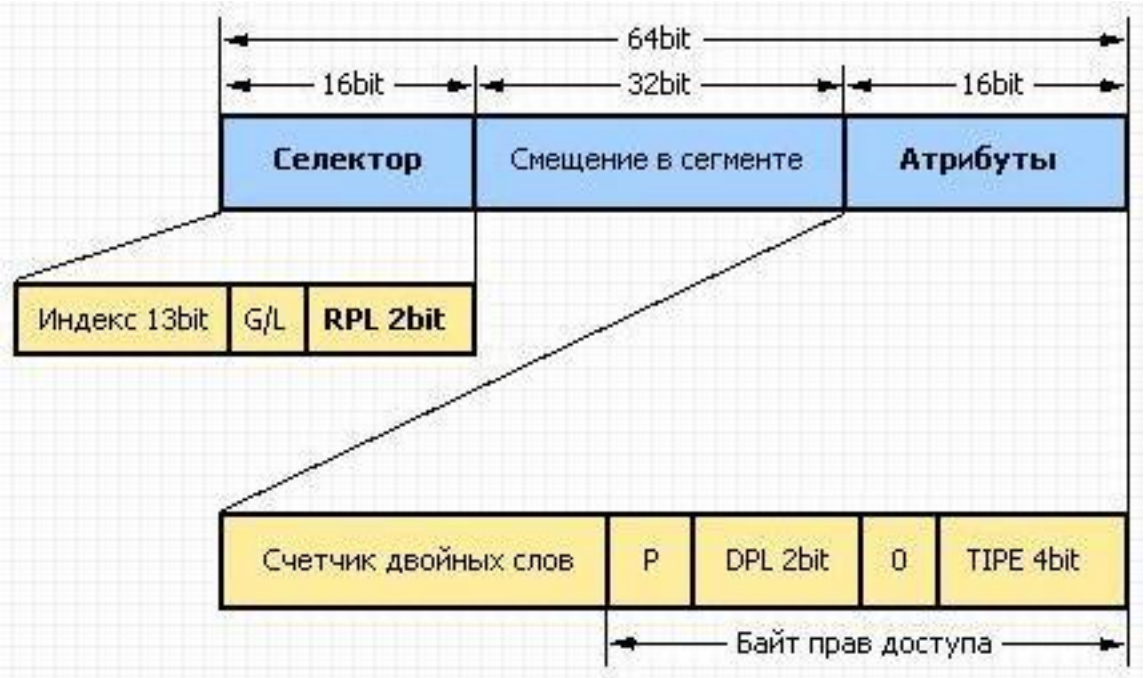


Рис. 4. Структура дескрипторов шлюзов.

2.2.2.2. Контрольные поля сегментных регистров и таблиц страниц.

Кроме дескрипторов для контроля и управления механизм защиты использует:

1. **Поле уровня привилегий запроса** (RPL) в селекторе команд загрузки селекторов или команд межсегментных переходов, Используется для проверки доступности соответствующего дескриптора. После загрузки селектора и дескриптора в соответствующие регистры поля уровней привилегий запроса (RPL) сегментных регистров данных, включая стековый, механизмом защиты уже не используются.
2. **Поле текущего уровня привилегии** (CPL - Current Privilege Level или Code Privilege Level) в сегментном регистре CS (кода). Используется при всех проверках механизма защиты по привилегиям.
3. **Двухразрядное поле уровня привилегий ввода/вывода** (IOPL) регистра флагов (EFLAGS). Определяет уровень привилегий программы по использованию команд ввода/вывода.
4. **Поля пределов таблиц виртуальной памяти** в каталоге разделов. Содержат размеры таблиц страниц и используются для контроля выхода за пределы границ таблиц.
5. **Поле предела каталога разделов виртуальной памяти** в управляющем регистре CR3. Содержат размеры каталога разделов и используются для контроля выхода за пределы границ каталога разделов.
6. **"Мандатные" поля** таблиц каталога разделов и таблиц страниц виртуальной памяти. Содержат:
   - бит уровня защиты привилегий пользователь/система (U/S) страницы или таблицы страниц;
   - бит прав доступа чтение/запись (R/W) страницы или таблицы страниц;
   - трехразрядное поле для организации дополнительной защиты операционной системой.
7. **Указатель битового поля разрешения ввода/вывода** в сегменте состояния задачи TSS. Определяет устройства ввода/вывода, к которым индивидуально разрешено обращение из данной задачи.

Вопросы для самопроверки:

1. *Где берет механизм запроса поле уровня привилегий запроса (RPL) при загрузке селекторов или выполнении команд межсегментных переходов.*
2. *Где хранится текущий уровень привилегий CPL*
3. *Где хранится поле уровня привилегий ввода/вывода IOPL*
4. *Что определяет поле IOPL*
5. *Где хранятся поля пределов таблиц страниц виртуальной памяти.*
6. *Где хранится указатель битового поле разрешения ввода/вывода.*
7. *Что определяют элементы битового поля разрешения ввода/вывода*
8. *Какие биты содержат "мандатные" поля строк таблиц каталогов и страниц.*

2.3. УРОВНИ ПРИВИЛЕГИЙ.

2.3.1. Концепция уровней привилегий.

Термин "привилегии" подразумевает права или возможности, которые обычно не разрешаются. Введение "неравноправия" программ в виде уровней привилегий является средством защиты кодовых сегментов и сегментов данных операционной системы. Защищаются программы операционной системы различных уровней иерархии от ошибок в пользовательских программах и программах операционной системы более низких уровней иерархии.

В многопрограммных режимах используется централизованное автоматическое распределение ресурсов. С этих позиций управляющие программы операционных систем могут рассматриваться как сервисные, предоставляющие пользовательским программам услуги по использованию ресурсов. Пользователи лишаются возможности непосредственно использовать команды управления системой, а для корректной реализации каждой разрешенной процедуры управления они могут использовать соответствующие программы операционной системы.

В свою очередь операционные системы имеют свою иерархическую структуру, свои логические уровни управления. Программы более высокого логического уровня управления активно используют программы более низкого уровня. На самом нижнем уровне управления находятся программы управления конкретными физическими устройствами (программы BIOS).

Таким образом, операционные системы изначально имеют свои уровни управления, где каждый логический уровень является сервисным для программ более высокого уровня. При этом, чем ниже логический уровень программ, тем большими возможностями по управлению они обладают. Естественно, что сбои в этих программах могут привести к более тяжелым последствиям и степень их защищенности должна быть выше.

Уровни привилегий обычно изображаются в виде колец защиты, показанных на рис. 5. Внешние кольца защиты соответствуют более высокому логическому уровню управления, но наименьшему уровню защиты.

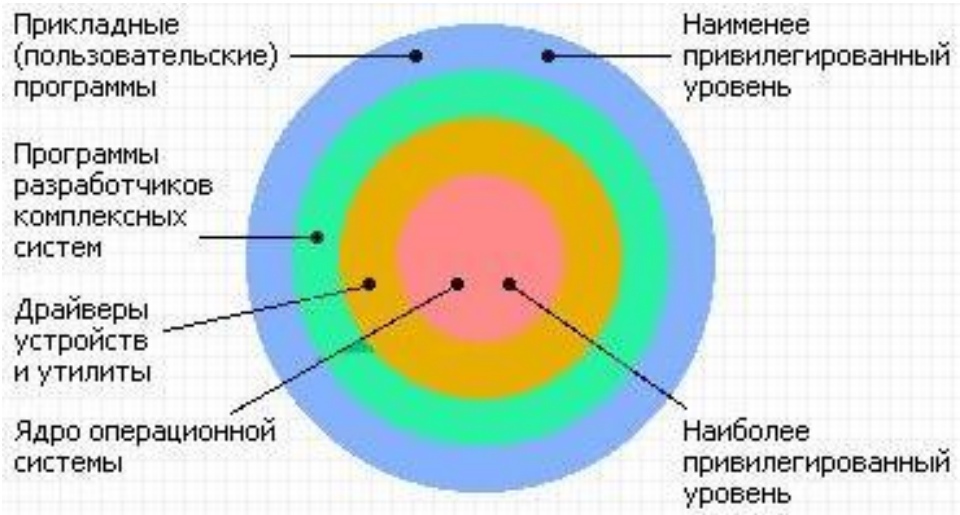


Рис. 5. Уровни привилегий и кольца защиты.

Механизм защиты МП x8б поддерживает до четырех уровней привилегий. Количество реализованных уровней привилегий определяется используемой операционной системой. Операционная система необязательно должна поддерживать все четыре уровня привилегий.

Простую незащищенную систему можно целиком реализовать в одном уровне привилегий. Традиционные системы супервизор/пользователь работают с двумя уровнями: супервизору (т.е. операционной системе) назначается уровень 0, а пользователям - уровень 3; примером такой системы может служить операционная система UNIX. Операционная система OS/2 поддерживает три уровня: код операционной системы работает на уровне 0, прикладные программы выполняются на уровне 3 и специальные процедуры для обращения к устройствам ввода-вывода действуют на уровне 2.

Четыре уровня привилегий МП ix86 обозначаются номерами: 0,1,2 и 3. Чем меньше номер уровня, тем меньше его логический уровень управления, но тем более он привилегирован и имеет большую степень защиты. Уровень 0 является наиболее, а уровень 3 - наименее привилегированными.

На рис. 5 нулевой уровень привилегий присвоен ядру операционной системы. Первый уровень предназначен для драйверов устройств и утилит. Второй уровень - для программ разработчиков комплексных систем. Третий уровень - для программ пользователей.

Проверка защиты по уровням привилегий осуществляется при выполнении почти каждой машинной команды во время работы МП в защищенном режиме (Р-режиме). МП в Р-режиме постоянно следит за тем, чтобы текущая программа была достаточно привилегированна для:

- выполнения некоторых (привилегированных) команд;
- обращения к данным других программ;
- передачи управления внешнему (по отношению к самой программе) коду командами передачи управления типа FAR.

Строго говоря, в МП ix86 реализованы две системы защиты по уровням привилегий. Вторая (дополнительная) система реализована в рамках механизма трансляции страниц и рассматривается в разд. 2.4.2.3.

Вопросы для самопроверки:

1. *Сколько уровней привилегий предусмотрено в МП ix86.*
2. *По какому критерию распределяются программы по уровням привилегий.*
3. *Какой уровень привилегий по номеру соответствует наибольшему уровню по защищенности.*
4. *Корректность каких трех действий программы контролируется по уровням привилегий.*

2.3.2. Задание уровней привилегий.

Основными объектами механизма защиты по привилегиям являются сегменты кодов и данных, включая стековые, и шлюзы. Именно им назначаются уровни привилегий. Уровень привилегий сегмента или шлюза определяет поле DPL (Descriptor Privilege Level) уровня привилегий, который находится в байте AR прав доступа соответствующего дескриптора. Это относится как к данным, так и к командам.

Эквивалентом понятия уровня привилегий процессора в МП ix86 является понятие текущего уровня привилегий CPL (Current Privilege Level или Code Privilege Level).

Текущей уровень привилегий определяется полем DPL дескриптора кодового сегмента. После загрузки кодового сегмента и передачи на него управления значение DPL копируется в поле CPL регистра кодового сегмента CS. После этого уровень привилегий кодового сегмента DPL становится текущим уровнем привилегий CPL. Таким образом, текущий уровень привилегий является уровнем привилегий исполняемого кодового сегмента.

Кроме уровней привилегий DPL и CPL, механизм защиты использует понятие уровня привилегий запроса (RPL). Это уровень привилегий, заданный селектором команды загрузки регистров сегментов данных, межсегментных передач управления, шлюзов.

Уровень привилегий запроса (RPL) селекторов сегментов данных сохраняется в соответствующих сегментных регистрах SS, DS, ES, FS, CS. После загрузки селекторов в сегментные регистры, хранящийся в них RPL **механизмом защиты уже не используется.** При межсегментных передачах управления селектор целевых сегментов кодов загружается в сегментный регистр CS, но вместо поля RPL заносится CPL=DPL дескриптора кодового сегмента.

Понятие уровня привилегий запроса RPL служит для защиты программ и данных операционной системы от ошибок программ, изменивших уровень привилегий при межуровневых передачах управления.

Вопросы для самопроверки:

1. *Каким объектам назначаются уровни привилегий.*
2. *Какое поле определяет уровень привилегий объекта. Где оно расположено.*
3. *Что определяет и где расположено поле текущего уровня привилегий.*
4. *Что определяет и где расположено поле уровня привилегий ввода - вывода.*
5. *Как формируется текущий уровень привилегий.*
6. *Как задается уровень привилегий запроса.*

2.4. РАБОТА МЕХАНИЗМА ЗАЩИТЫ.

2.4.1. Проверки корректности использования команд.

Привилегированные команды. К привилегированным командам относятся те, на выполнение которых влияет уровень привилегий программы или привилегии устройств ввода/вывода.

МП ix86 имеет 3 группы привилегированных команд степени привилегий которых различны:

- PL0 - команды;
- IOPL - "чувствительные" (IOPL-sensitive) команды;
- команды, модифицируемые в соответствии с текущим уровнем привилегии.

PL0-команды. Это команды, выполнение которых разрешено только на уровне привилегий 0. При попытке выполнить их в Р-режиме на другом уровне привилегий генерируется сигнал нарушения общей защиты (прерывание 13).

В некоторых литературных источниках только эти команды определяются как привилегированные.

В зту группу команд входят команды останова процессора и команды загрузки системных объектов и передачи данных, в которых источником или получателем данных выступают системные регистры управления CRn, отладки DRn и проверки TRn:

- *HLIT - останов процессора;*
- *CLTS - сброс флажка переключенной задачи;*
- *LGDT - загрузка регистра глобальной таблицы дескрипторов;*
- *LIDT - загрузка регистра таблицы дескрипторов прерываний;*
- *LLDT - загрузка регистра локальной таблицы дескрипторов;*
- *LTR - загрузка регистра задачи;*
- *LMSW - загрузка слова состояния машины;*
- *MOV GRn, CRn;*
- *MOV GRn, DRn;*
- *MOV GRn, TRn;*
- *MOV CRn.GRn;*
- *MOV DRn, GRn;*
- *MOV TRn.GRn.*

IOPL - "чувствительные" команды. Это команды, которые изменяют состояние флажка прерываний IF, выполняют захват шины или операцию ввода-вывода.

В группу IOPL - "чувствительных" команд входят команды:

- *CLI запрещение прерываний (IF := 0);*
- *STI разрешение прерываний (IF := 1),*
- *IN, INS ввод данных из входного порта;*
- *OUT, OUTS вывод данных в выходной порт;*
- *LOCK захват шины. Команда префикс. Используется в многопроцессорных системах для захвата шины другим процессором.*

Для выполнения этих команд программа необязательно должна иметь уровень привилегий 0. Привилегированность этих команд заключается в том, что их могут выполнять программы, уровень привилегий которых выше уровня, определяемого полем IOPL уровня привилегий ввода-вывода в регистре EFLAGS. Другими словами, для выполнения этих команд требуется, чтобы CPL был численно меньше или равен IOPL (CPL <= IOPL). При попытке выполнить их в P-режиме при CPL > IOPL генерируется нарушение общей защиты (прерывание 13).

Команды обращения к устройствам ввода/вывода (IN, INS, OUT, OUTS) являются особой подгруппой IOPL - "чувствительных" команд. На их выполнение, кроме соотношения уровней привилегий CPL <= IOPL, влияет содержимое битовой карты разрешения ввода/вывода в сегменте состояния задачи TSS.

Выполнение команд ввода/вывода разрешено, если CPL <= IOPL. При CPL > IOPL производится проверка содержимого битовой таблицы разрешения ввода/вывода.

Вначале проверяется корректность самой таблицы. Таблица должна заканчиваться байтом из всех единиц (FF).

Каждый бит в таблице разрешает или запрещает обращение к однобайтовому порту. При обращении к многобайтовому порту проверяется разрешение для каждого байта порта. Разрешающим значением бита является - нулевое значение.

При несоблюдении разрешающего соотношения уровней привилегий и отсутствии разрешения по битовой таблице обращения хотя бы к одному байту порта в P-режиме генерируется нарушение общей защиты (прерывание 13).

Командами, модифицируемыми в соответствии с текущим уровнем привилегии, являются две команды:

- POPFD - загрузка регистра EFLAGS четырехбайтным значением из стека;
- POPF - загрузка регистра FLAGS двухбайтным значением из стека.

Команды POPFD и POPF сами по себе не являются привилегированными, но их выполнение зависит от значения CPL, т.е. от уровня привилегий содержащего их кода. Здесь речь идет о защите по модификации полей регистра флагов IOPL и IF.

Флаг IOPL может модифицироваться только PL0-программами. По сути дела IOPL является PL0-флагом.

Флаг IF может модифицироваться IOPL-чувствительными командами при CPL <= IOPL. В этом смысле IF можно считать IOPL-чувствительным флагом.

Любая программа может содержать команды POPFD и POPF. Эти команды могут изменять любые биты флагов, но биты IOPL могут быть изменены этими командами, только если выполняется PLO-программа, а флаг IF - если выполняется условие CPL <= IOPL

Процессор никак не сообщает об изменении действия этих команд. Это вполне штатная ситуация. Процессор просто не модифицирует биты IOPL и флаг IF, если это не разрешено.

Поясним особенности выполнения этих команд на конкретных примерах.

Поскольку поле IOPL находится не в дескрипторе, а в более доступном регистре FLAGS, может показаться, что способ защиты ввода-вывода проработан не очень хорошо. Предположим, что прикладная программа безуспешно попыталась выполнить команду ввода-вывода. Тогда можно записать копию регистра EFLAGS в стек, преобразовать его как любой операнд в памяти, а затем извлечь из стека модифицированный образ в регистр флагов:

pushfd *; Включить флажки в стек*

or dword ptr ss:[esp], 3000h *; Задать IOPL= 3*

popfd *; Извлечь флажки из стека*

; Теперь можно производить ввод-вывод

Однако такой прием в МП ix86 не работает из-за особенностей выполнения команд POPFD и POPF.

Примерно такая же ситуация характерна для модификации флага прерываний IF. Команды сброса CLI и установки STI этого флага являются IOPL-чувствительными. Если оказалось, что прямое их выполнение невозможно, можно попытаться аналогичным способом воздействовать на флажок IF регистра EFLAGS:

pushfd *; Включить флажки в стек*

and dword ptr ss:[esp],Offffffdffh *; Сбросить IF*

popfd*; Вернуть флажки*

; Прерывания запрещены

Но это тоже не проходит. Команда popfd просто не изменит ни IOPL, ни IF.

Вопросы для самопроверки:

1. *Что такое PL0-команды.*
2. *Какие команды входят в группу PL0 - команд.*
3. *Чем заканчиваются попытки некорректного использования PL0 - команд.*
4. *Какие команды входят в группу IOPL - чувствительных команд.*
5. *В чем заключается привилегированность IOPL - чувствительных команд.*
6. *В чем заключается привилегированность команд обращения к устройствам ввода/вывода.*
7. *Какие команды входят в группу команд модифицируемых в соответствии с текущим уровнем привилегии.*
8. *В чем заключается особенность команд модифицируемых в соответствии с текущим уровнем привилегии.*

2.4.2. Защита данных.
2.4.2.1. Стратегия защиты данных.

Все прикладные программы в многопрограммном режиме выполняются на одном (нижнем) уровне привилегий. Механизм виртуальной памяти обеспечивает их защиту от взаимных помех.

Для сегментов данных системных программ, дескрипторы которых размещены в глобальной таблице дескрипторов, также возможно разделение адресных пространств, но только за счет механизма трансляции страниц. Поэтому всем программам разрешено обращаться к любым данным на своем уровне привилегий.

Логика сервисных функций современных операционных систем предусматривает работу с данными программ менее защищенных режимов. Поэтому программам разрешено обращаться к данным и на менее привилегированных уровнях.

Защита программ от взаимных помех здесь определяется только корректностью написания системных программ

Программам не разрешается считывание/запись элементов данных, которые имеют более высокий уровень привилегий, т.е. "движение" к данным внутрь колец защиты запрещается. Любая такая попытка приводит к фиксации нарушения общей защиты.

Это общее правило защиты доступа к данным можно записать в виде условия: CPL (т.е. PL программы) <= DPL (т.е. PL данных) или представить графически в виде допустимых схем обращения к данным (рис. 6.).

На рис. 6 уровни привилегий представлены в виде горизонтальных "этажей", сегменты программ - в виде треугольников, данных - в виде квадратов. Стрелками без перекрещивания указаны разрешенные схемы доступа.

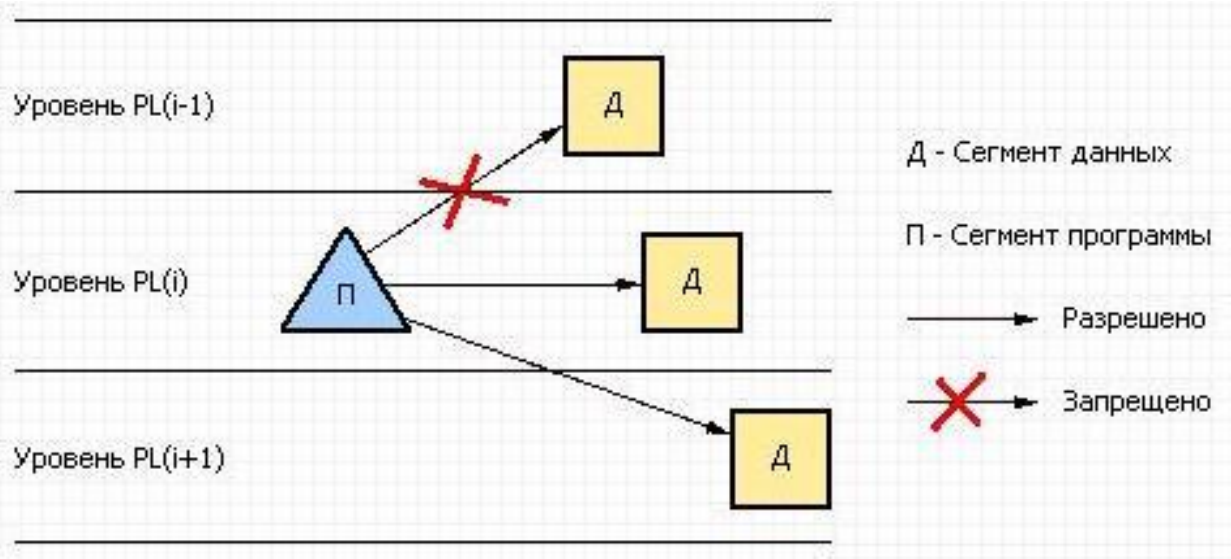


Рис. 6. Схемы допустимых обращений к данным.

Представленное общее правило защиты доступа к данным является "рамочным", отражающим только максимальные возможности доступа. Полное правило доступа несколько сложнее.

Дело в том, что программы менее защищенных уровней, используя сервисные программы более защищенных уровней, могут выполнять процедуры на уровне этих сервисных программ. В общем случае, это могут быть процедуры модифицирования данных, заданных косвенно через передаваемые параметры и расположенных на более защищенном уровне привилегий. Возникает опасность несанкционированного нарушения данных на более защищенном уровне привилегий.

Для блокирования этой опасности механизм защиты использует более сложную схему проверки условий доступа.

В условие проверки вводится параметр "эффективный уровень" привилегий (Effective Privilege Level - EPL) как функция уровня привилегий запросчика RPL:

$$EPL = \max [CPL, RPL]$$

Параметр EPL используется в условии проверки разрешения доступа к сегментам вместо CPL.

С учетом RPL условие проверки разрешения доступа к данным приобретает вид:

$$EPL <= DPL \text{ или}$$

$$\max [CPL,RPL] <= DPL \text{ или}$$

$$\underline{(CPL<=DPL)\&(RPL<=DPL)}$$

Из приведенных выражений для условия доступа ясно, что значение RPL может только уменьшить круг допустимых по обращению к данным уровней привилегий.

Например:

1. *При (CPL = RPL = 2) и (EPL = 2) - программе доступно обращение к данным на втором и третьем уровнях привилегий.*
2. *При (CPL = 2) и (RPL = 3) - значение EPL = 3 и программе доступно обращение к данным только на третьем уровне привилегий.*

При RPL < CPL значение EPL всегда равно CPL, т.е. при помощи уменьшения RPL нельзя увеличить круг допустимых по обращению к данным уровней привилегий. Подбором значения RPL при необходимости можно только ослабить возможности текущего уровня привилегий по доступу к данным. В этом заключается основное предназначение параметра RPL

При загрузке нового сегмента данных в команде загрузки указывается, непосредственно или косвенно, селектор сегмента данных, который содержит значение RPL Пользователь, программирующий на уровне ассемблера, имеет возможность манипулировать значением RPL. Но при любом значении RPL он не сможет задать EPL <= (CPL = 3).

Процедуры более высоких уровней привилегий пишутся системными программистами. Для системных программистов RPL является средством управлять доступом к данным на системных уровнях в соответствии с общей сер-

висной стратегией конкретной операционной системы. Программа может получить доступ к данным на более высоком уровне привилегий, только если он предусмотрен системными программистами и только через системные процедуры.

Защита данных различных программ на одном уровне привилегий производится на основе разделения адресных пространств механизмом виртуальной памяти.

В механизме трансляции сегментов предусмотрены локальные таблицы дескрипторов. Локальные таблицы через соответствующие селекторы привязываются операционной системой к программам определенной задачи. Сегменты, заданные дескрипторами локальной таблицы, "видимы" только для программ, к которым они привязаны.

При использовании механизма трансляции страниц разделение адресных пространств задач производится при помощи индивидуальных таблиц каталогов и страниц.

Следует отметить, что оба механизма трансляции при разделении адресных пространств допускают возможность создания общих массивов данных (сегментов или страниц) совместно используемыми (разделяемыми) различными программами.

Кроме проверок на соответствие уровней привилегий при обращении к данным производятся и общие проверки:

- на выход за пределы сегмента и таблиц,
- соответствие типа сегмента,
- прав использования содержимого сегмента или страницы.

Вопросы для самопроверки:

1. *В чем заключается общее правило обращения к данным.*
2. *Для каких целей используется уровень привилегии запроса.*
3. *Как производится разделение адресных пространств программ одного уровня привилегий при сегментной организации памяти.*
4. *Как производится разделение адресных пространств программ одного уровня привилегий при страничной организации памяти.*

4.2.2. Работа механизма защиты данных при трансляции сегментов.

В работе механизма защиты обращений к данным можно выделить два этапа:

- загрузка сегмента данных;
- обращение к сегменту данных для чтения или записи операндов при выполнении команд программы.

Загрузка сегментов данных (включая стековые) производится командами "Загрузка указателя в регистр смещения". Это команда типа регистр - память:

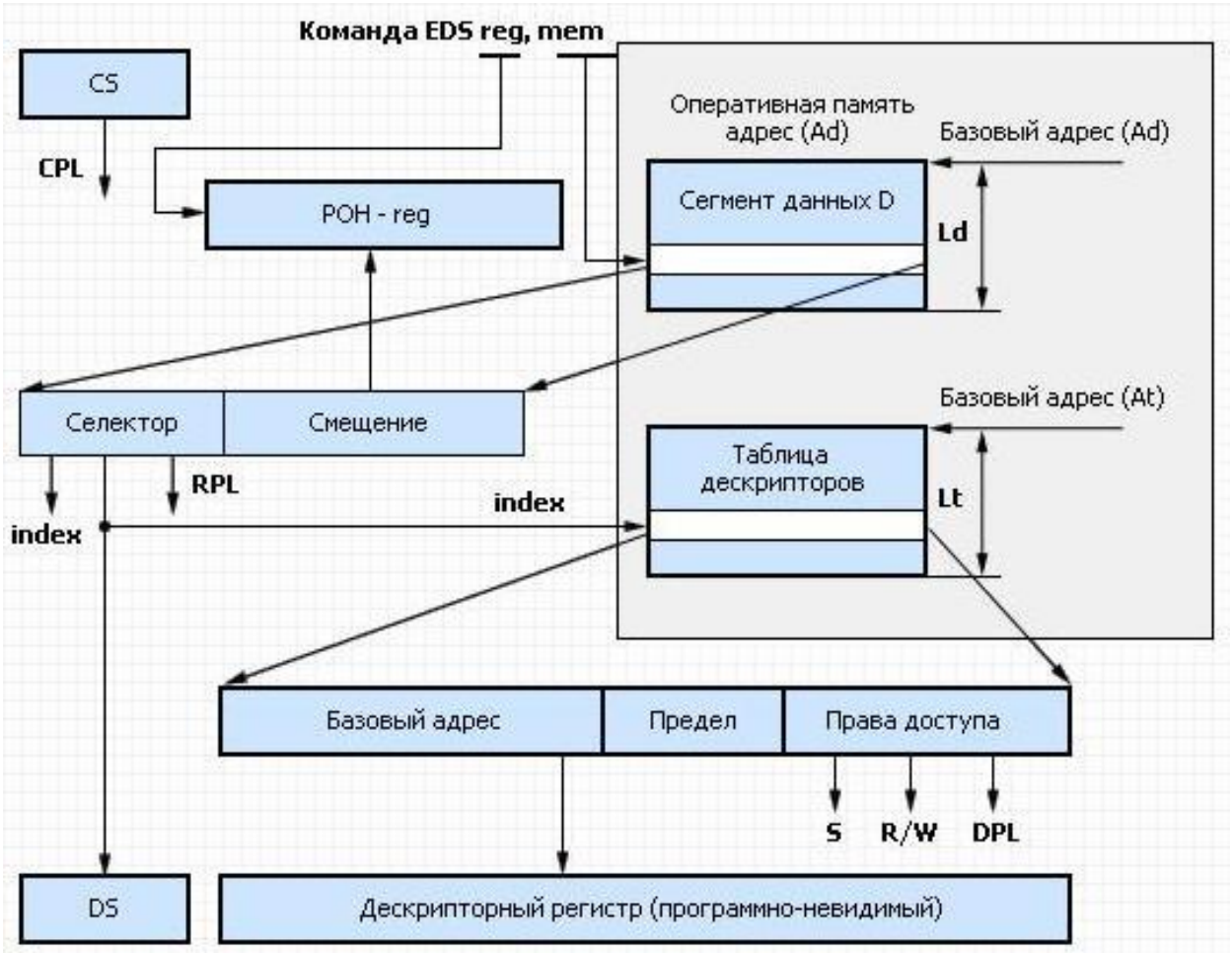
(LPS, LES, LFS, LGS, LSS) reg, mem

Сегментный регистр определяется кодом операции. В адресной части команд полем "reg" задается один из регистров общего назначения, полем "mem" - адрес полного указателя операнда: смещение в сегменте и селектор сегмента.

Команда выбирает из ячеек памяти, начиная с адреса mem, величину смещения некоторого операнда и селектор. При положительных результатах проверок механизма защиты смещение загружается в РОН, заданный полем "reg", а селектор - в сегментный регистр, заданный полем mem. Загрузка селектора в сегментный регистр сопровождается чтением из таблицы дескрипторов (локальной или глобальной в зависимости от значения бита G/L) дескриптора сегмента, заданного полем index селектора и записи его в дескрипторный регистр, ассоциированный с заданным сегментным регистром. Непосредственно эта команда не предусматривает каких-либо действий по использованию выбранного смещения.

Со схемой выполнения команды загрузки сегментного регистра можно познакомится или по рисунку (рис. 7) или по экранной модели (мод. 1 – см. электронный учебник).

Схема выполнения команды загрузки сегментного регистра.



Проверки по соответствию уровней привилегий:
для команд (LDS, LES, LFS, LGS.) reg, mem - (CPL <= DPL) & (RPL <= DPL);
для команды LSS peg, mem. - (CPL = DPL = RPL)

Рис.7. Схема выполнения команд загрузки указателей в регистры сегментов

Основными этапами выполнения команды и проверок являются:

- *Чтение по адресу тет смещения в сегменте данных DS и селектора целевого сегмента. Селектор целевого сегмента используется для обращения к таблице дескрипторов. Выбранные значения смещения и селектора временно сохраняются в буферных регистрах.*
- *Обращение к таблице дескрипторов. Перед обращением к таблице дескрипторов проводится ряд проверок:*
    1. *Действительность селектора целевого сегмента. Он должен быть не нуль-селектором (только при обращении к GDT). Проверка производится по полю index (рис. 7). Нуль-селектор определяет нулевой индекс. Условие проверки index <> 0.*
    2. *Возможность выхода за пределы таблицы дескриптора. Индекс селектора не должен превышать предел таблицы дескрипторов index <= Lt, где Lt- предел таблицы дескрипторов. При этом выбор таблицы определяется битом типа таблицы (G/L) селектора. Базовый адрес таблицы дескрипторов At и предел Lt определяются полями регистра таблиц дескрипторов GTDR или LDTR.*
- *Обращение к таблице и чтение дескриптора целевого сегмента. После чтения дескриптора целевого сегмента (сегмент временно сохраняется в буферном регистре) проверяются:*
    1. *Тип дескриптора по байту прав доступа. Дескриптор должен быть несистемным (S = 0) и читаемым (R = 1).*
    2. *Условие разрешения доступа к данным по уровням привилегий (CPL <= DPL) & (RPL <= DPL)*

Из приведенных выражений для условия доступа ясно, что значение RPL может только уменьшить круг допустимых по обращению к данным уровней привилегий.

Например:

1. *При (CPL = RPL = 2) и (EPL = 2) - программе доступно обращение к данным на втором и третьем уровнях привилегий.*
2. *При (CPL = 2) и (RPL = 3) - значение EPL = 3 и программе доступно обращение к данным только на третьем уровне привилегий.*

При RPL < CPL значение EPL всегда равно CPL, т.е. при помощи уменьшения RPL нельзя увеличить круг допустимых по обращению к данным уровней привилегий. Подбором значения RPL при необходимости можно только ослабить возможности текущего уровня привилегий по доступу к данным. В этом заключается основное предназначение параметра RPL.

При загрузке нового сегмента данных в команде загрузки указывается, непосредственно или косвенно, селектор сегмента данных, который содержит значение RPL. Пользователь, программирующий на уровне ассемблера, имеет возможность манипулировать значением RPL. Но при любом значении RPL он не сможет задать EPL <= (CPL = 3).

Процедуры более высоких уровней привилегий пишутся системными программистами. Для системных программистов RPL является средством управлять

доступом к данным на системных уровнях в соответствии с общей сервисной стратегией конкретной операционной системы. Программа может получить доступ к данным на более высоком уровне привилегий, только если он предусмотрен системными программистами и только через системные процедуры.

Для всех сегментов данных (кроме стековых) при выполнении указанных условий процессор выполняет следующие действия:

- *значение селектора загружается в соответствующий сегментный регистр (на рис. 7 - DS);*
- *значение дескриптора целевого сегмента данных загружается в дескрипторный регистр, ассоциатированный с заданным сегментным регистром;*
- *значение смещения загружается в РОН, указанный полем reg команды.*

В противном случае процессор формирует нарушение общей защиты и не производит никаких изменений в выбранном сегментном регистре данных.

Обращение к сегменту данных для чтения или записи операндов при выполнении команд программы.

После успешной загрузки селектора, при выполнении команд, предусматривающих обращение к памяти за данными, процессор производит проверки:

- Соответствия адреса смещения в сегменте пределу сегмента данных. Адрес mem не должен превышать предел сегмента данных (mem <= Ld).
- Разрешенность запрашиваемой операции (считывание или запись) для этого сегмента (проверяется по биту R/W поля прав доступа дескриптора сегмента)

  На этом этапе обнаруживаются и отвергаются попытки записи в сегменты кода или в только считываемые сегменты данных, а также считывание из сегмента кода, для которого разрешено только выполнение. Очевидно, что такие ситуации невозможно выявить при загрузке селектора, так как процессор не может заранее знать о возможном неправильном использовании сегмента.

При загрузке селектора в сегментный регистр стека процессор несколько ужесточает правило защиты: загрузка разрешается, если только значение DPL сегмента стека точно равно значению CPL и RPL. Это связано с использованием отдельных стеков для каждого уровня привилегий.

Кроме этого, загрузка селектора в регистр SS производится, если только выбираемый сегмент допускает операции считывания и записи, а также присутствует в памяти (бит P присутствия в дескрипторе сегмента должен находиться в состоянии 1). Это связано с тем, что стек участвует в сохранении состояния процессора при передачах управления программам обработки прерывания, и недоступность стека по записи и/или чтению парализует работу системы прерывания.

Последующие проверки по обращению к стеку с операциями чтения или записи не производятся, так как необходимый контроль был произведен при загрузке селектора.

Механизм защиты допускает возможность размещения данных (констант) непосредственно в кодовом сегменте. Это может быть желательным при хранении кода и данных в ПЗУ. Обращение к этим данным (только по чтению) возможно при помощи использования префикса переопределения сегмента, определенному как "читаемый".

Вопросы для самопроверки:

1. *Определите этапы в работе механизма защиты при обращении к данным загрузка сегмента данных.*
2. *Какие проверки производятся перед обращением к таблице дескрипторов.*
3. *Как определяется действительность селектора целевого сегмента.*
4. *Как определяется возможность выхода за пределы таблицы дескриптора.*
5. *Какие проверки производятся после чтения дескриптора целевого сегмента.*
6. *Что означает проверка по типу сегмента.*
7. *В чем заключается условие разрешения доступа к данным по уровням привилегий.*

2.4.2.3. Работа механизма защиты данных при трансляции страниц.

Механизм трансляции страниц предоставляет дополнительные возможности по защите обращений к данным. Они включают разделение адресных пространств данных различных задач и ряд дополнительных компонент защиты.

В отличие от трансляции сегментов, здесь используются более мелкие по размерам объекты памяти: разделы и страницы, составляющие сегмент. Доступ к этим объектам производится через строки каталога PDE *(Page Directory Entry)* и строки таблиц страниц PTE *(Page Table Entry).*

Строки каталога PDE и таблиц страниц PTE по своим функциям являются дескрипторами этих объектов и, кроме адресных полей и пределов, содержат поля прав доступа. Каждое поле прав доступа (рис. 8) содержит биты:

- уровня защиты привилегий пользователь/система (U/S);
- прав использования - чтение/запись (R/W);
- трехразрядное поле для организации дополнительной защиты операционной системой.

Бит U/S. Позволяет операционной системе ограничивать доступ к данным страниц, например, при размещении своих констант в страницах пользователя. При значении бита U/S = 0 доступ к данным программ пользователя запрещен по чтению и по записи.

Бит R/W. Позволяет ограничивать доступ по типу использования данных. При R/W=1, разрешен доступ по чтению и по записи, при R/W = 0 - только по чтению.

Трехразрядное поле для организации дополнительной защиты операционной системой. Являются резервом ОС.

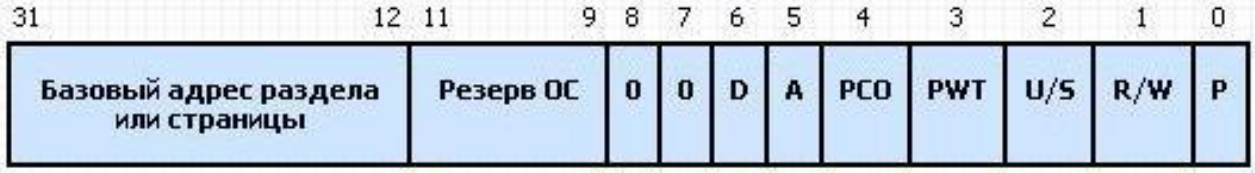


Рис. 8. Поля строки каталога (раздела) или страницы.

Запреты, устанавливаемые битами U/S и R/W каталога PDE и таблицы PTE, распространяются на программы только третьего уровня привилегий, причем, биты U/S и R/W каталога PDE определяют доступ ко всем страницам раздела, а биты U/S и R/W таблицы PTE - к странице.

Доступ к данным сегмента определяется объединением запретительных ограничений прав доступа трех уровней: дескриптора сегмента, строки каталога и строки таблицы страниц.

Вопросы для самопроверки:

1. Какие поля прав доступа содержат строки каталога и таблиц страниц.

2. Что определяет бит прав использования.

3. На какие программы распространяются запреты полей прав доступа строк каталога и таблицы страниц.

4. Как определяются по совокупности проверок права доступа к данным: сегмента, каталога и страницы.

2.4.3. Защита программ.

2.4.3.1. Стратегия защиты программ при межсегментных переходах.

Программы в мультипрограммной среде наряду с данными могут разделять и процедуры, например, библиотеки подпрограмм. С другой стороны, для локализации программных ошибок в пределах программы необходима и защита. Стратегии защиты программ и защиты данных имеют много общего, но имеются и существенные различия.

Защита как программ, так и данных на одном уровне привилегий производится на основе разделения адресных пространств механизмом виртуальной памяти при помощи локальных таблиц дескрипторов, каталогов и страниц. При обращении к программам также производятся проверки на выход за пределы сегмента и таблиц, проверки соответствия типа сегмента и проверки прав использования содержимого сегмента или страницы. Для совместного использования программных кодов применяются или общие страницы, или общие сегменты.

Различия в защите программ и данных проявляются при межсегментных передачах управления с изменением уровней привилегий. Эти различия определяются особенностями механизма смены кодовых сегментов в МП Intel и моделями взаимодействия программ с разными уровнями привилегий.

Особенности механизма смены кодовых сегментов в МП Intel.

В МП ix86 нет команд загрузки кодового сегмента CS и указателя ко-

манд IP. Содержимое IP устанавливается при начальной загрузке после включения или рестарта МП, увеличивается (продвигается) при выборке очередной команды в команде и произвольно изменяется при выполнении команд перехода, вызова процедур, процедур прерывания или возврата. Содержимое CS устанавливается при начальной загрузке и меняется при выполнении команд межсегментного перехода, вызова процедур, процедур прерывания или возврата. В МП Intel для защищенного режима предусмотрены два механизма перехода:

- *передача управления,*
- *переключение задач (смена контекста).*

Оба механизма перехода предусматривают возможности перехода с возвратом (переход на подпрограмму).

Причинами перехода могут быть команды передачи управления, прерывания и ловушки.

Модели взаимодействия программ.

Для обеспечения эффективного взаимодействия программных продуктов процессор предоставляет пользователям определенный набор средств организации взаимодействия программ:

- *команды условных или безусловных переходов (JMP);*
- *команды перехода на подпрограмму (переход с возвратом CALL);*
- *команды возврата из подпрограммы (RET);*
- *прерывания и ловушки, включая команды прерывания;*
- *команды возврата из подпрограммы обработки прерывания (IRET).*

Механизм защиты в определенных случаях ограничивает набор возможных средств организации взаимодействия программ. Эти ограничения касаются только межсегментных переходов (переходов типа FAR). При внутрисегментных переходах производятся только обычные проверки на выход за пределы сегмента.

С точки зрения функционирования механизма защиты можно выделить несколько моделей взаимодействия программ. Модели различаются типами и средствами реализаций межсегментных переходов.

Механизм защиты различает следующие типы межсегментных переходов:

1. *использующие процедуры передачи управления и переключения задач;*
2. *с возможностью и без возможности возврата;*
3. *на подпрограмму и возврата из подпрограмм;*
4. *с изменением и без изменения уровня привилегий;*
5. *на обычные и подчиненные кодовые сегменты.*

Комбинации указанных типов переходов образуют набор возможных моделей взаимодействия программ.

Механизм защиты, используя систему проверок (включая проверки по уровням привилегий взаимодействующих программ), определяет текущую модель взаимодействия программ и допустимые средства взаимодействия.

Основными постулатами, определяющими стратегию функционирования механизма защиты, являются:

1. *программы и данные на одном уровне привилегий можно достаточно надежно защитить от взаимных помех за счет разделения адресных пространств средствами механизма виртуальной памяти;*
2. *сервисные программы всегда относятся к программам операционной системы. Они расположены на не менее привилегированном уровне, чем программы, использующие их услуги;*
3. *переход программ на более высокий уровень привилегий увеличивает опасность нарушения защиты и должен производится под жестким контролем операционной системы;*
4. *попытки передачи управления с уменьшением уровня привилегий, кроме возврата из подпрограммы и переходов с использованием механизма переключения задач, считаются ошибочными и недопустимыми;*
5. *возврат из (сервисной) подпрограммы или программы обработки прерывания без использования механизма переключения задач производится только командами RET или IRET;*
6. *при использовании механизма переключения задач допускаются переходы с уменьшением уровня привилегий с использованием команд CALL и JMP.*

Для упорядоченного рассмотрения алгоритмов работы механизма защиты все возможные способы организации переходов условно представим в следующей структуре:

1. Межсегментные передачи управления.
   - без изменения уровней привилегий (к подчиненным сегментам и неподчиненным),
   - без изменения или с увеличением уровня привилегий,
   - без изменения или с уменьшением уровня привилегий.
2. Межсегментные переключения задач.

   В этой структуре все возможные способы организации переходов делятся на две основные группы по используемому механизму переходов:
   - *передачи управления;*
   - *переключения задач.*

   В переходах с использованием механизма передачи управления различаются три группы. В названиях этих групп отражены возможности изменения уровня привилегий, причем для всех групп предусмотрена возможность перехода без изменения уровня привилегий. Основные различия между организациями переходов в указанных группах заключаются в ограничениях по использованию средств переходов и особенностях организации защиты. Особенностью межсегментных передач управления без изменения или с увеличением уровня привилегий является обязательное использование "шлюзов" и ограничение в использовании команды JMP.

Межсегментные передачи управления без изменения или с уменьшением уровня привилегия - это средства возврата из процедур.

Вопросы для самопроверки:

1. *Определите возможные механизмы межсегментных переходов.*
2. *Назовите команды переходов и другие средства организации переходов.*
3. *Определите основные постулаты, определяющие стратегию функционирования механизма защиты (6 постулатов).*

2.4.3.2. Межсегментные передачи управления без изменения уровня привилегий.

Эти переходы предусматривают использование команд CALL и JMP.

В этой модели дается полная свобода в организации взаимодействия программ. При передачах управления возможно задание в командах любого целевого адреса. Но это чревато и возможными ошибками. Например, проектировщик программы может указать ошибочный целевой адрес или даже адрес не первого байта команды. В последнем случае гарантирована непредсказуемая работа процессора (с порчей данных и "зависанием"). Поэтому контроль на допустимость такой работы более жесткий, чем при обращении к данным. Такие взаимодействия программ допускаются только в пределах своего уровня привилегий.

Существуют две разновидности моделей межсегментных переходов без изменения уровня привилегий. Это:

- переходы на программы подчиненных сегментов,
- переходы на программы сегмента того же уровня привилегий.

Переходы на программы подчиненных сегментов.

Среди сервисных программ операционной системы имеется ряд программ, типа преобразования двоичных целых чисел в символьный код ASCII, которые используют другие программы на каждом уровне привилегий. Обычно такие программы просты по конструкции, требуют обращения только к своему параметру и возвращают результат вызывающей программе. Для упрощения обращения к таким программам можно было бы размещать их на уровне пользовательских программ. Но их используют на всех уровнях. Следовательно, их нужно или дублировать для каждого уровня привилегий, или использовать более "громоздкие "межсегментные переходы без изменения или с увеличением уровня привилегий".

Для таких ситуаций в процессоре Intel предусмотрено альтернативное решение - использование подчинённых кодовых сегментов.

Кодовый сегмент определяется как подчиненный установкой бита C - Conforming в байте прав доступа AR дескриптора кодового сегмента. 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 кода, который передает им управление с помощью команд CALL или JMP. 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 кодового сегмента в нем не должны содержаться привилегированные команды, например, команды ввода-вывода.

Когда управление передается подчиненному сегменту кода, биты поля CPL регистра CS не принимают значение поля DPL дескриптора нового сегмента кода, как это обычно бывает, а сохраняют значение DPL последнего выполнявшегося неподчиненного сегмента кода. Таким образом, переход на программы подчиненного кодового сегмента является переходом **без изменения текущего уровня привилегий.**

Но даже для подчиненных кодовых сегментов имеются ограничения по их использованию. Программы подчиненных сегментов не передают свой уровень вызывающей программе, но все же имеют собственный уровень, определяемый полем DPL дескриптора сегмента.

Передавать управление подчиненному сегменту может программа, уровень привилегий которой не выше уровня привилегий целевого кодового сегмента, т.е. если DPL <= CPL. Чтобы программа подчиненного сегмента была доступна на всех уровнях привилегий, она должна иметь высший уровень привилегий ($DPL_{ЦКС} = 0$). Это вполне соответствует общему правилу использования сервисных программ: любая программа может быть сервисом только для программ более низкого уровня иерархии.

Механизм защиты при использовании команд межсегментных переходов проверяет бит C прав доступа дескриптора целевого кодового сегмента. При обнаружении признака подчиненности сегмента производится проверка на соответствие уровням привилегий: DPL <= CPL

При выполнении команд межсегментных передач управления изменяются регистры CS и EIP, поэтому, кроме проверки подчиненности целевого кодового сегмента, производится проверка достоверности селектора, загружаемого в регистр CS.

Переходы на программы кодовых сегментов того же уровня привилегий.

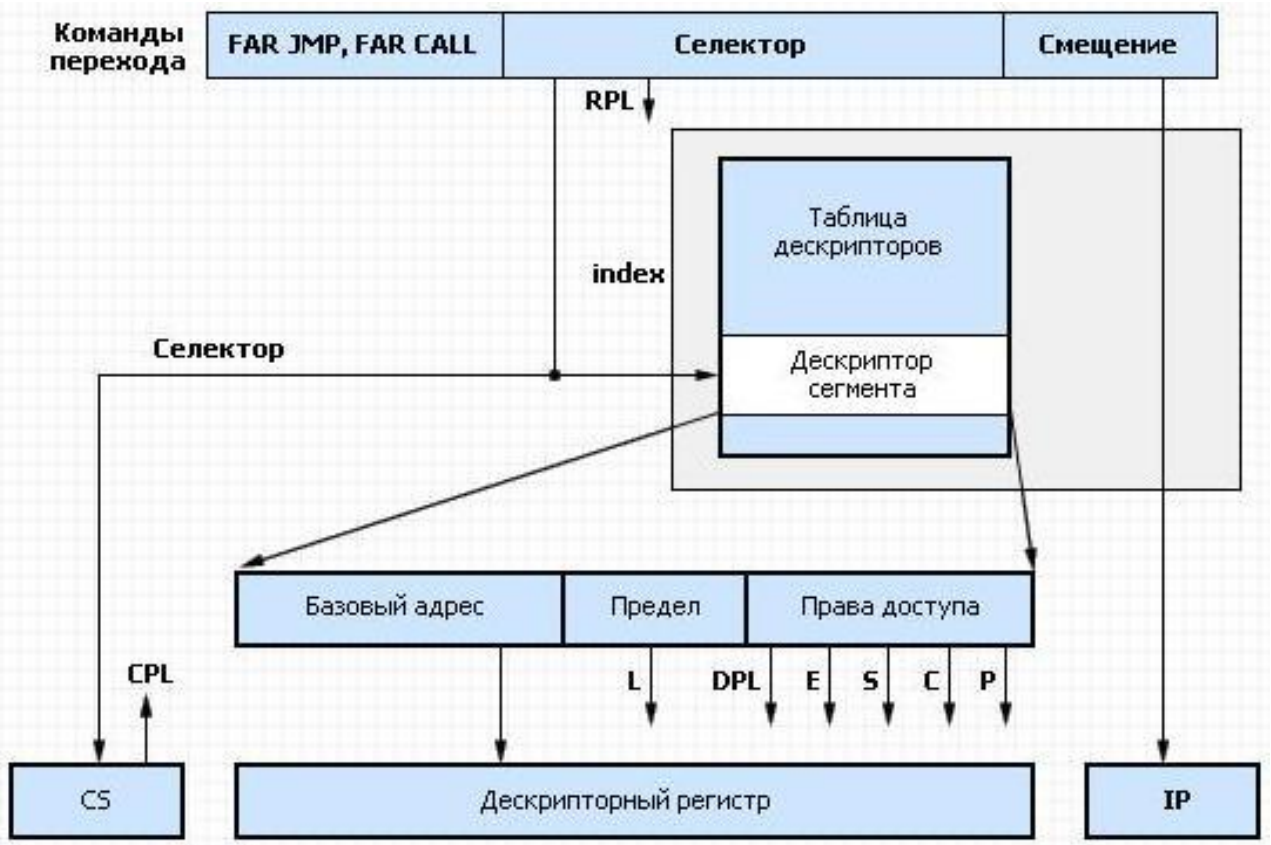
Условием использования этой модели взаимодействия программ является (CPL = DPL) & (RPL <= CPL).

Кроме проверки по уровням привилегий, как и при использовании подчиненных сегментов, делаются "штатные" проверки достоверности селектора целевого кодового сегмента, загружаемого в регистр CS.

Со схемой взаимодействия программ без изменения уровня привилегий можно ознакомится или по рисунку (рис. 9), или по экранным моделям (мод. 2 (переход на подчиненные сегменты) – см. электронный учебник, мод. 3 (переход на не подчиненные сегменты) – см. электронный учебник).

При загрузке селектора в регистр CS процессор производит ряд проверок.

На рис. 9 представлены только основные проверки, отражающие стратегию контроля механизма защиты.



Проверки:

1. *(S=0) (E=l) = 1 - проверка на исполняемость*
2. *C=1 - проверка на подчиненность сегмента*
3. *Если C=0 (не подчиненный), то (CPL = DPL) & (RPL <= CPL) - проверка соответствия уровней привилегий*
4. *(P=1) - проверка на присутствие сегмента*
5. *Смещение <= L - проверка на обращение в пределах сегмента*

Рис.9. Схема проверок и межсегментных переходов без изменения уровня привилегий с использованием команд и механизма передач управления

Прежде всего, проверяется, определяет ли целевой дескриптор кодовый сегмент, т.е. имеет ли атрибут исполняемого сегмента (S=0, E=1). Разрешение на считывание из этого сегмента не требуется.

Далее проверяется подчиненность целевого кодового сегмента (C=1).

В случае использования неподчиненного целевого кодового сегмента процессор контролирует, что значение DPL дескриптора точно равно значению CPL, а значение поля RPL в селекторе команды меньше или равно значению CPL (рис. 9).

Наконец, целевой кодовый сегмент должен быть отмечен как присутствующий (бит P = 1 в дескрипторе, рис. 9), а новое значение для EIP должно находиться в пределах нового сегмента кода (смещение <= предела L, рис. 9).

Только при условии выполнения всех этих условий процессор продолжает выполнение передачи управления по указанному адресу. Когда какая-либо проверка дает отрицательный результат, формируется нарушение общей защиты (прерывание 13) или нарушение присутствия (прерывание 11).

В последнем случае программа обработки прерывания производит загрузку целевого сегмента в оперативную память, устанавливает бит присутствия. После этого возобновляется выполнение передачи управления.

Рассмотренная модель взаимодействия программ дает максимум возможностей проектирования, проста по реализации, но имеет минимум защиты программы от возможных ошибок.

Вопросы для самопроверки:

1. *Какие команды допустимы для межсегментных передач управления без применения шлюзов. Возможны ли при этом изменения уровней привилегий.*
2. *Какие программные ошибки исключают применение шлюзов.*
3. *Для использования каких программ применяются подчиненные сегменты.*
4. *Как задаются подчиненные сегменты.*
5. *Какое условие проверки по уровням привилегий определяет доступность подчиненного сегмента.*
6. *Какое условие проверки по уровням привилегий определяет доступность межсегментных передач управления без изменения уровней привилегий.*
7. *Какими проверками, кроме соответствия уровней привилегий, сопровождается процедура межсегментной передачи управления.*

2.4.3.3. Межсегментные передачи управления без изменения или с увеличением уровня привилегий.

Необходимость и проблемы межсегментных переходов без изменения или с увеличением уровня привилегий.

В процессоре Intel запрещена передача управления обычными средствами сегменту кода, находящемуся на другом уровне привилегий.

Это самый важный и самый сложный момент механизма защиты по привилегиям. Ограничивая передачу управления в пределах одного кольца защиты, процессор предотвращает произвольное изменение уровней привилегий. Если бы значение CPL можно было легко изменять, все остальные средства защиты оказались бы бессмысленными.

Но передача управления на программы операционной системы необходима.

Необходимость такой передачи связана с одной из главных функций операционной системы - быть посредником между пользовательскими программами и системой распределения и управления ресурсами ЭВМ. Операционная система MS-DOS только через прерывание INT 21h предоставляет пользовательским программам более 150 разнообразных сервисных процедур.

Рассмотренное применение подчиненных сегментов пригодно только для частных случаев использования программных кодов, не производит переходы с изменением уровня привилегий программного кода и не решает проблему в целом.

Для реализации фактического увеличения уровня привилегий при межсегментных переходах в МП корпорации Intel предусмотрены:

1. Особые системные объекты - шлюзы (gates).
2. Ограничения используемых средств переходов.
3. Изменения в интерпретации адресного поля команд переходов.
4. Особая стратегия проверок уровней привилегий.
5. Изменения в процедуре переходов.

Шлюзы.

Шлюзы являются ключевыми объектами для организации межсегментных переходов с увеличением уровня привилегий. Любые переходы с увеличением уровня привилегий производятся только с использованием шлюзов. Как альтернативный вариант с большим уровнем защиты, шлюзы могут использоваться и при межсегментных переходах без изменения уровня привилегий.

Дескриптор шлюза вызова действует как посредник или своеобразный "интерфейсный слой" между сегментами кода, находящимися на различных уровнях привилегий.

В отличие от сегментов, шлюзы ассоциированы не с массивом адресов, а с определенным целевым логическим адресом (полным указателем - селектор, смещение) передачи управления. При желании шлюз можно ассоциировать с меткой перехода. Исключением является шлюз задачи. Он ассоциируется не с логическим адресом, а с целевой задачей.

Шлюзы идентифицируют разрешенные точки (метки) в коде, которым может быть передано управление, и являются единственным средством межсегментной передачи управления с увеличением уровня привилегий. В этом смысле они являются обработчиками соответствующих переходов. По аналогии, когда передачи управления производятся без использования шлюзов и селектор в команде указывает не на дескриптор шлюза, а на дескриптор кодового сегмента или дескриптор TSS, говорят, что обработчиком перехода является кодовый сегмент или сегмент TSS.

Таким образом, разрешенные точки входа в сервисные программы задаются не вызывающей программой, а шлюзом. Программа не может читать шлюзы и определять адреса и расположение сервисных процедур. В этом смысле сервисные процедуры становятся "прозрачными" для вызывающих программ, а использование шлюзов - защитой от несанкционированного использования программного кода и "неверных" адресов передач управления, приводящих к "зависанию" программы.

Шлюзы имеют свои уровни привилегий. Уровень привилегий шлюза может отличаться от уровня привилегий как вызываемой программы, так и вызывающей. При вызове шлюза проверяется его доступность по соотношению уровней привилегий вызывающей программы и уровня привилегии шлюза.

Кроме этого, проверяется соотношение уровней привилегии вызывающей и вызываемой программы.

Используемые средства переходов.

Межсегментный переход с увеличением уровня привилегий может быть задан:

- командой межсегментного вызова CALL типа FAR;
- командой вызова процедуры прерывания INT n;
- прерыванием;
- ловушкой.

В соответствии с этим используются шлюзы:

- вызова (call gates);
- прерывания (interrupt gates);
- ловушки (trap gates).

Кроме этого, для переключения задач с изменением уровня привилегий используется шлюз задач (task gates). Межсегментные переходы с использованием переключения задач рассматриваются ниже в п. 2.4.3.5.

Как системные объекты шлюзы имеют собственные дескрипторы, которые размещаются в дескрипторных таблицах:

- шлюзы вызова - в GDT и LDT;
- шлюзы прерываний и ловушек - в IDT;
- шлюзы задач - в GDT и IDT.

При использовании шлюзов меняется интерпретация адресного поля команд переходов.

Интерпретация адресного поля команд, использующих шлюзы (вызова, прерываний) для межсегментных передач управления с увеличением уровня привилегий.

Такими командами являются: CALL типа FAR и команды вызова прерывания, например, INT n.

Адресное поле команды CALL при межсегментных переходах без изменения уровня привилегий прямо, через РОН или адресное поле задает полный указатель целевого адреса CS:EIP. При межсегментных переходах с увеличением уровня привилегий интерпретация адресного поля команды изменена: смещение полного указателя игнорируется, а селектор используется для выборки шлюза вызова, который и определяет CS:EIP.

Команды вызова процедуры прерывания, прерывания и ловушки всегда используют шлюзы. Тип прерывания (n) является индексом соответствующего шлюза в таблице IDT.

Стратегия проверок уровней привилегий при межсегментных переходах с увеличением уровней привилегий.

При реализации межсегментных переходов с увеличением уровня привилегий производятся две проверки:

- доступность шлюза;
- доступность кодового сегмента.

На рис. 10 представлена схема допустимых вызовов процедур с использованием шлюзов.

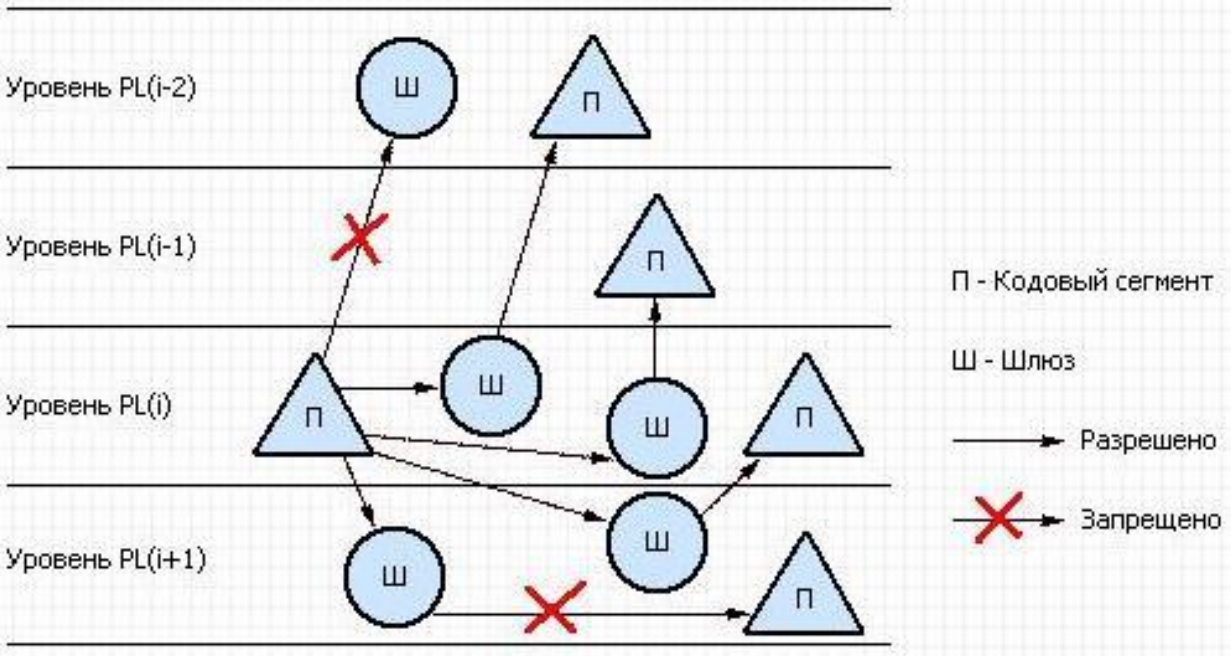


Рис. 10. Схема допустимых межсегментных переходов с использованием шлюзов.

Доступность шлюза.

Доступность шлюза аналогична доступности данных. Шлюз доступен, если его уровень привилегий ($DPL_{шл}$) не выше текущего уровня привилегий ($CPL_{пр}$) и уровня привилегий запроса ($RPL_{ск}$).

Для команды вызова проверка заключается в определении выполнения условия:

$$(CPL_{пр} <= DPL_{шл}) \ \& \ (RPL_{ск} <= DPL_{шл})$$

При использовании команды INT n, прерывания или ловушки отсутствует уровень запроса и проверка сводится к определению выполнения условия:

$$CPL_{пр} <= DPL_{шл}.$$

Доступность кодового сегмента.

Межсегментная передача управления допустима только на программы своего или более высоких уровней привилегий. В соответствии с этим доступность кодового сегмента определяется выполнением условия:

$$DPL_{цкс} <= CPL_{пр}$$

Таким образом, условиями допустимости передач управления с использованием шлюзов по уровням привилегий являются:

- При использовании команды CALL типа FAR:

  $(CPL <= DPL_{шл}) \ \& \ (RPL_{ск} <= DPL_{шл})$
  $DPL_{цкс} <= CPL$

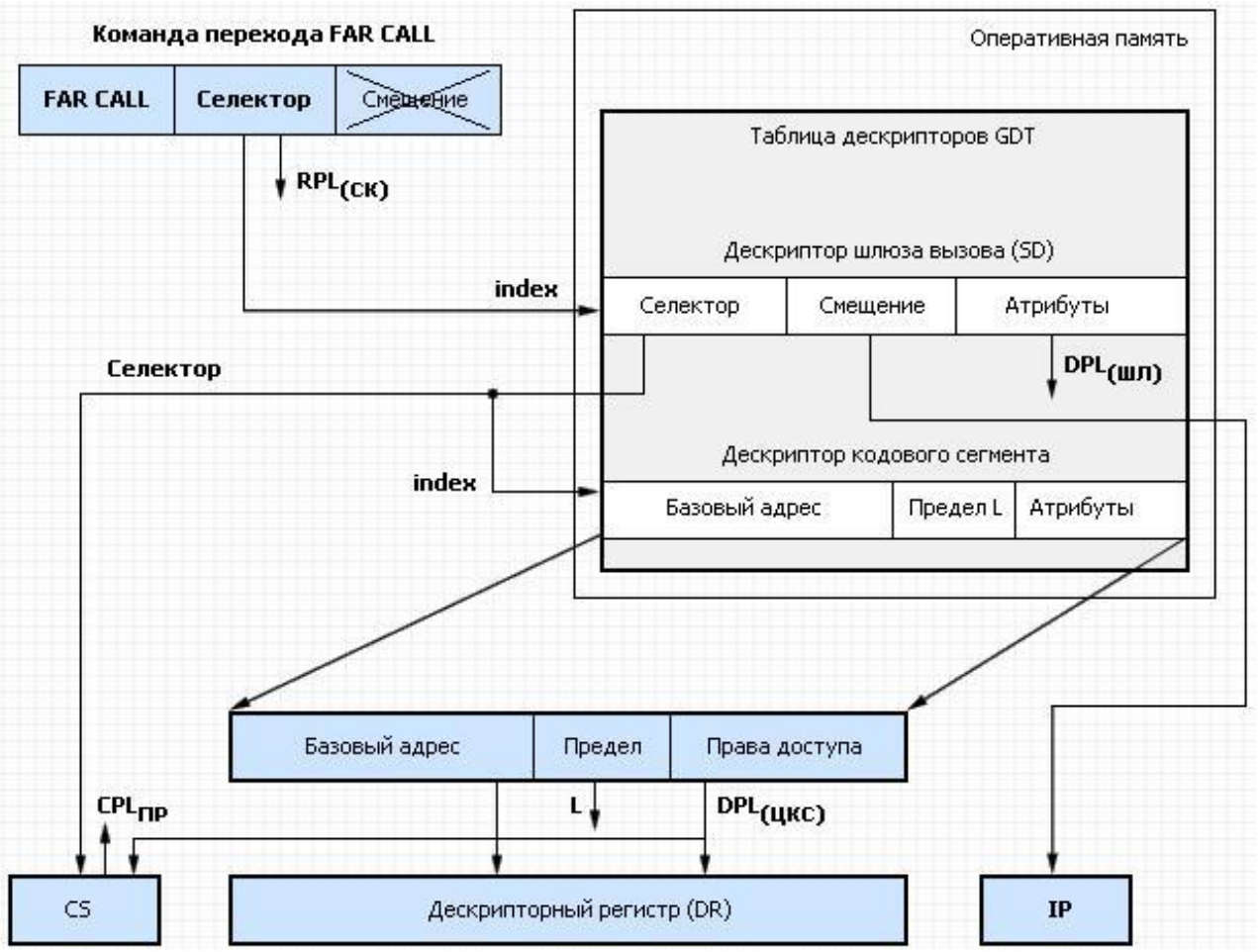
- При использовании команды INT n, прерывания или ловушки:

$CPL <= DPL_{шл}$
$DPL_{цкс} <= CPL$

Процедура межсегментных переходов с увеличением уровня привилегий.

Со схемой взаимодействия программ без изменения или с увеличением уровня привилегий по команде CALL типа FAR можно ознакомится или по схеме (рис. 11) или по экранной модели (мод. 4 – см. электронный учебник).

Обязательным условием межсегментных передач управления с увеличением уровня привилегий является использование шлюза вызова, заданного командой FAR CALL. Смещение, заданное командой, игнорируется.



Проверки привилегий по условиям доступа:

1. *($CPL_{пр} <= DPL_{шл}$) & ($RPL_{ск} <= DPL_{шл}$) - Проверка доступности шлюза вызова.*
2. *$DPL_{цск} <= CPL_{пр}$ - Проверка доступности целевого кодового сегмента*

Рис.11. Схема проверок и межсегментных переходов без изменения или увеличения уровня привилегий.

Чтение шлюза вызова является первым этапом выполнения команды. На этом этапе производятся стандартные проверки действительности селектора. Селектор шлюза вызова может указывать на глобальную или локальную таблицы дескрипторов.

После чтения дескриптора шлюза вызова проверяется соответствие его типа и условие его доступа:

$$(CPL <= DPL_{шл}) \& (RPL <= DPL_{шл})$$

При положительных результатах проверок выполняется второй этап перехода. Это чтение дескриптора целевого кодового сегмента, которое производится по селектору заданного в дескрипторе шлюза вызова. Перед чтением делаются стандартные проверки: на действительность селектора, на соответствие смещения в сегменте пределу сегментного кода и на доступность целевого кодового сегмента по уровням привилегий.

Проверка доступности целевого кодового сегмента по уровням привилегий определяется условием:

$$DPL <= CPL$$

При положительных результатах этой проверки производится третий этап перехода. При этом:

- *полный указатель адреса возврата (содержимое регистра CS указателя команд IP) сохраняется в стеке;*
- *дескриптор целевого кодового сегмента, выбранный из таблицы дескрипторов по селектору шлюза вызова, заносится в дескрипторный регистр (DR на рис. 11);*
- *значение DPL из селектора целевого кодового сегмента заносится в поле уровня привилегий (CPL) сегментного регистра CS;*
- *поле индекса и типа таблицы селектора целевого кодового сегмента из дескриптора шлюза вызова (SD на рис. 11) заносится в соответствующие поля сегментного регистра CS;*
- *значение поле смещения дескриптора шлюза вызова (SD на рис. 11) заносится в указатель команд IP.*

Если переход привел к изменению уровня привилегий, то производится еще один завершающий этап. Это смена стека.

В защищенном режиме для каждого уровня привилегий имеется свой стек.

Наличие индивидуальных стеков для каждого уровня привилегий диктуется двумя основными причинами:

- *вызываемая процедура должна защищаться от возможного переполнения стека, возникающего, если вызывающая программа распределила недостаточное стековое пространство;*
- *сегмент стека, используемый более привилегированной процедурой, может быть разрушен менее привилегированными программами; разделение же стека на привилегированных уровнях производится не программами пользователей, а системными процедурами, которые только используются прикладными программами.*

Начальные значения селектора стека SS и указателя ESP хранятся во всех сегментах состояния задачи TSS и не модифицируются. При смене стека адрес возврата CS и IP, селектор SS и указатель ESP старого стека сохраняются в новом стеке. В него же, в случае вызова процедуры с передачей пара-

метров, из старого стека копируются передаваемые параметры и счетчик двойных слов, определяющий их количество.

При смене стека производится обращение к сегменту TSS для чтения значений SS, загрузка нового указателя стека в регистр ESP и селектора стека в сегментный регистр SS. Процессор анализирует поле счетчика WC (Word Count) дескриптора шлюза вызова. Затем копирует из старого стека в новый число двойных слов, указанное этим полем. Это - параметры, передаваемые вызываемой процедуре. Они размещаются в новом стеке также, как в старом. Наконец, в новый стек включается адрес возврата CS и EIP.

Смена стека сопровождается стандартными проверками на действительность селектора сегмента стека и доступность сегмента стека по уровням привилегий перед его загрузкой в сегментный регистр стека SS. Дополнительно проверяется достаточность выделенного объема стека для размещения следующих элементов:

- старого указателя стека (8 байт);
- копируемых параметров (0-128 байт);
- адреса возврата (8 байт);
- локальных переменных, которые может включать в стек вызванная процедура.

Рассматриваемая модель передач управления с использованием шлюзов предполагает переходы как с увеличением, так и без изменения уровней привилегий.

Для передач управления без изменения уровня привилегий это более сложный (и медленный) способ перехода по сравнению с обычным способом - без применения шлюзов. Но есть ряд факторов, по которым использование шлюзов является целесообразным при переходах и без изменения уровня привилегий.

1. *Не все сервисные процедуры, используемые на разных уровнях привилегий, можно пометить как подчиненные. Например, из-за того, что они содержат привилегированные команды. Но это программы общего использования и их дескрипторы сегментов должны помещаться в глобальных таблицах. Основным средством защиты от взаимных помех для таких программ остается использование шлюзов.*
2. *Использование шлюзов не только делает "прозрачными" сервисные программы, но и является удобным интерфейсным слоем между программами. Программа-потребитель должна "знать" только селектор дескриптора шлюза.*

Если не происходит изменения уровня привилегий, то для передачи управления возможно использование не только команды вызова процедуры, но и команды межсегментной передачи управления без сохранения адреса возврата JMP типа FAR. В этом случае используется только заданный командой селектор дескриптора шлюза вызова, а смещение, заданное командой, игнорируется. При переходе адрес возврата не сохраняется.

Передачи управления по прерываниям и ловушкам.

Передачи управления по прерываниям и ловушкам являются так же передачами без изменения или с увеличением уровня привилегий. Это тоже передачи управления с обязательным использованием шлюзов прерывания или ловушки (прерывания с использованием шлюза задачи рассматриваются в п 2.4.3.5).

Отличиями этих переходов от вызовов процедур командой CALL типа FAR являются следующие:

- команды вызова прерывания, например, INT n, прерывания и ловушки используют не селектор шлюза, а тип прерывания n;
- обращение за дескрипторами шлюзов производится не в таблицы GDT или LDT, а в таблицу дескрипторов прерываний IDT.

Использование типа прерывания, вместо селектора дескриптора шлюза, изменяет условие доступа. **Команды прерывания, прерывания и ловушки для определения целевого кодового сегмента используют не селектор, а тип прерывания.** Для прерываний и ловушек дескриптор шлюза доступен, если его уровень привилегий не превышает текущего, т. е.:

$$CPL <= DPL_{шл}$$

Условие доступа целевого кодового сегмента остается прежним. Целевой кодовый сегмент доступен программам, если его уровень не меньше текущего:

$$DPL_{цкс} <= CPL$$

Со схемой взаимодействия программ с увеличением уровня привилегий по команде INT n, прерываниям или ловушкам можно ознакомится по экранной модели (мод. 5 – см. электронный учебник).

Вопросы для самопроверки:

1. *В чем заключается необходимость передач управления с повышением уровня привилегий.*
2. *Что предусмотрено в МП корпорации Intel для реализации переходов с увеличением уровня привилегий.*
3. *Какой обработчик может использоваться при переходах с увеличением уровня привилегий.*
4. *Можно ли использовать шлюзы при переходах без изменения уровня привилегий.*
5. *Можно ли использовать команду FAR CALL для передач управления с увеличением уровня привилегий.*
6. *Какие средства можно использовать для передач управления с увеличением уровня привилегий.*
7. *Как задается смещение и целевой сегмент при передачах управления с увеличением уровня привилегий.*
8. *Какие типы шлюзов вы знаете.*
9. *Где могут располагаться шлюзы.*

10. В чем заключается условие доступности шлюза вызова.

11. В чем заключается условие доступности шлюзов прерывания, ловушки.

12. В чем заключается условие доступности целевого сегмента.

13. При каких переходах производится смена стека.

14. По каким причинам производится смена стека.

2.4.3.4. Межсегментные передачи управления без изменения или с уменьшением уровня привилегий.

Это модель взаимодействия программ при возврате из вызываемых процедур или программ обработки прерываний или ловушек.

Передачи управления при возвратах производятся командами: из вызываемых процедур – RET, RET n или из программ обработки прерываний (ловушек) - IRET.

Вторая форма команды возврата RET n производит удаление из стека n байт. Она необходима в том случае, когда при вызове процедуры передавались параметры (содержимое поля счетчика двойных слов WC в шлюзе вызова не равно нулю).

Передача управления на программы обработки прерывания (ловушки) всегда производится без уменьшения уровня привилегий. Соответственно, возврат из них всегда должен быть без увеличения уровня прерывания. Попытки возврата с увеличением уровня привилегий для этих случаев считаются ошибкой.

Переходы при вызове процедур, прерывания или ловушки могут использовать механизм передач управления или механизм смены задач. При возврате из процедуры всегда используется тот же механизм перехода.

Возврат из подпрограмм на основную программу по командам RET, RET n и IRET не использует шлюзы. Если процедура вызова использовала механизм передачи управления, то обработчиком перехода является кодовый сегмент. Выполняемые при возврате действия аналогичны действиям при вызове, но в обратном порядке:

- *из стека процедуры извлекается адрес возврата CS:EIP;*
- *если выполняется команда RET n, то производится инкремент регистра ESP на n для удаления параметров из стека процедуры;*
- *из стека процедуры извлекается указатель старого стека SS:ESP.*

Схемы возврата и проверок такие же, как при выполнении перехода по командам FAR CALL или FAR JMP (передача управления без изменения уровня привилегий).

Отличиями являются:

- команды возврата не содержат адреса переходов;
- полный указатель точки возврата (селектор, смещение) берется из стека, а не из команды;
- изменено условие корректности перехода;
- при изменении уровня привилегий производится восстановление ста-

рого стека.

В обычных условиях указатель возврата всегда достоверен, поскольку он был сформирован командой CALL или INT. Но здесь учитывается возможность того, что вызываемая процедура могла изменить указатель или неправильно обработать стек. Проверки при возврате из процедур направлены на исключение таких ситуаций.

При выполнении возврата из процедур переход считается корректным, если уровень привилегии селектора целевого кодового сегмента, извлеченного из стека, совпадает с уровнем привилегии дескриптора этого сегмента и не превышает текущий уровень привилегий, т.е.:

$$(CPL <= DPL_B) \& (DPL_B = CPL_B),$$

где:

DPL_B - уровень привилегии сегмента возврата (из стека),

CPL_B - уровень привилегии селектора сегмента возврата (из таблицы дескрипторов).

Кроме проверок соответствия уровней привилегий, при загрузке селектора кодового сегмента и селектора сегмента стека (восстановление стека) выполняются все стандартные проверки.

Со схемой взаимодействия программ без изменения или с уменьшением уровня привилегий командами возврата: из вызываемых процедур - RET, RET n или из программ обработки прерываний (ловушек) - IRET ознакомится по экранной модели (мод. 6 – см. электронный учебник).

Вопросы для самопроверки:

1. *В каких случаях допускаются передачи управления с уменьшением уровня привилегий.*
2. *Какими средствами производятся передачи управления с уменьшением уровня привилегий.*
3. *Какой механизм перехода используется при организации возврата из процедур.*
4. *Какой обработчик используется при возврате из процедуры.*
5. *Как определяется точка возврата из процедуры.*
6. *Определите условие доступности возврата из процедуры по уровням привилегий.*

2.4.3.5. Межсегментные переключения задач.

Межсегментные переключения задач - это переходы с использованием механизма переключения задач.

Модели переключения задач могут использоваться:

- *для переключения независимых программ при параллельном их выполнении в режиме разделения времени;*
- *для вызова процедур;*
- *для вызова программ обработки прерываний и ловушек;*

- *для 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. Это системные сегменты, предназначенные для копирования основных регистров процессора, хранящих "контекст" программы. Иногда переключение задач называют сменой контекста. Термин "задача" здесь означает "выполняемая программа", вернее - "программа, находящаяся на стадии выполнения". В многопрограммном (многозадачном) режиме работы в стадии выполнения могут находиться несколько программ. Для каждой из них создается сегмент состояния задачи - TSS. Выполнение этих программ может производиться одним процессором в режиме разделения времени. Основным назначением механизма переключения задач является организация очередных переходов между выполняемыми программами.

При переключении программ содержимое процессора (контекст программы) копируется в TSS текущей программы, а содержимое TSS целевой программы переписывается в регистры процессора. Такой переход исключает прямое взаимодействие переключаемых программ.

Возможны две модели переключения задач:

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

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

Переключение задач.

При переключении задач по прерываниям и ловушкам дескрипторы шлюзов задач или TSS должны находиться в таблице дескрипторов прерываний IDT. В случае использования команды CALL типа FAR сохраняется адрес возврата, но параметры не передаются.

С точки зрения взаимодействия программ, переключение программ равнозначно завершению одной программы с сохранением на магнитном диске данных и запуску следующей программы. Но 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.

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 *служат для обработки некоторого внешнего события, явно не связанного с прерываемой программой. Для этих случаев целесообразно ис-*

пользовать переключение задач. Выход из подпрограмм (обратное переключение) производится командой 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, например, INT n, или прерыванием, то процессор устанавливает бит вложенности NT в регистре флагов FLAGS и бит занятости B в поле прав доступа дескриптора. Затем записывает селектор дескриптора TSS прерываемой программы в специальное поле "селектора возврата" TSS принимающей программы. 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.

Процедура обработки прерывания при каждом вызове должна начинаться с начального адреса. Но при выходе сохраняется адрес, следующий за командой выхода IRET. Эта проблема может быть решена чисто программным путем. Для возможности использования вызываемой процедуры переключением задач процедура должна быть зациклена и начальная команда процедуры должна следовать непосредственно или через безусловный переход за командой 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.

Прямое переключение задач.

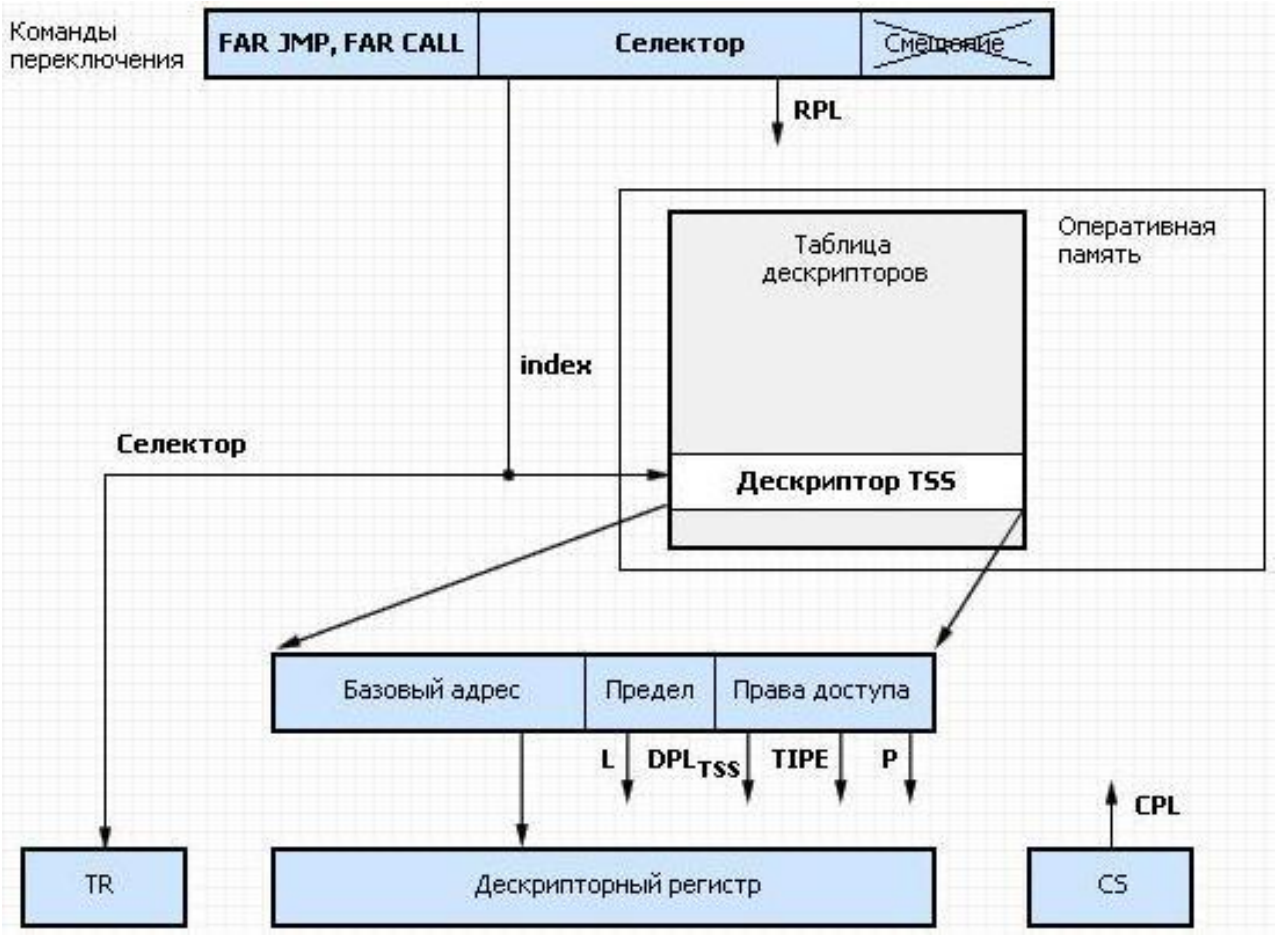
Прямое переключение задач - это переключение задач без использования шлюза задач. Прямое переключение задач производится командами JMP и CALL типа FAR .

В случае использования команд JMP и CALL типа FAR селекторы этих команд должны выбирать дескриптор TSS и должно выполняться условие доступа прямого переключения: уровень привилегий TSS программы цели должен быть не выше уровня привилегий текущей программы и запроса, т.е.:

$$(CPL <= DPL_{TSS}) \ \& \ (RPL <= DPL_{TSS}),$$

где DPL_{TSS} - уровень привилегий сегмента TSS (берется из поля уровня привилегий дескриптора TSS).

Со схемой основных проверок при прямом переключении задач по командам JMP и CALL типа FAR можно ознакомиться или по схеме (рис. 12) или по экранной модели (мод. *7* – см. электронный учебник).



Проверки:

1. *CPL <= DPL_{TSS}? - Проверка соответствия уровней привилегий*
2. *Тип сегмента - TSS? - Проверка соответствия типа сегмента*
3. *P=1? - Проверка на присутствие сегмента*
4. *Индекс <= L? - Проверка на обращение в пределах сегмента*

Рис.12. Схема основных проверок и процедура выбора целевого TSS при прямом переключении задач

Перед обращением к таблице дескрипторов проверяется действительность селектора и соответствие индекса границам таблицы дескрипторов. После выборки дескриптора TSS проверяется его тип, присутствие TSS в оперативной памяти и его доступность по уровням привилегий:

$$(CPL <= DPL_{TSS}) \ \& \ (RPL <= DPL_{TSS})$$

При положительных результатах проверок производится процедура переключения задач (сохранение содержимого регистров процессора в старом TSS и загрузка селектора целевого TSS в регистр TR).

Загрузка селектора целевого TSS приводит к загрузке регистров процессора контекстом целевой задачи, т.е. к переключению задач.

Прямое переключение задач допускает переключение на любой уровень привилегий, но используется только для переключений без увеличения уровня привилегий. Программно TSS доступен только для переключения задач.

При прямом переключении задач проверяется уровень привилегии не дескриптора кодового сегмента, а дескриптора сегмента состояния задачи. Уровень привилегии дескриптора определяет не уровень привилегий целевого кодового сегмента, а уровень привилегий программ, которым разрешено производить переключение задач на целевой кодовый сегмент, т.е. уровень доступности процедуры переключения задач. Но по чтению и по записи TSS программно недоступен. Создание, модификацию и чтение TSS операционные системы могут производить путем отображения в тот же адрес памяти дескриптора сегмента данных.

В условие корректности прямого переключения программ не входит проверка уровня привилегий (DPL дескриптора) целевого кодового сегмента. Следовательно, прямое переключение задач, даже по команде JMP типа FAR, может производиться и на программы с более низким уровнем привилегий. Возможно и переключение на задачи с более высоким уровнем привилегий. Но это - именно переключение программ, а не уменьшение или увеличение уровней привилегий программы. Старая задача сохранилась в старом TSS, и началось выполнение новой программы со своим уровнем привилегий.

Тем не менее, для переключения на программу с более высоким уровнем привилегий предусмотрено использование шлюза задачи (защищенное переключение задач). Это, вероятно, связано не столько с проблемами защиты, сколько с унификацией форм сервисных процедур.

Основное назначение переключения задач является организацией многозадачных режимов работы. Это прерогатива операционной системы. Дескрипторы TSS хранятся в глобальной таблице дескрипторов GDT и имеют высокий уровень привилегий. Поэтому прямое переключение задач производится, фактически, на меньший уровень привилегий.

Переключение задач с использованием шлюза задач.

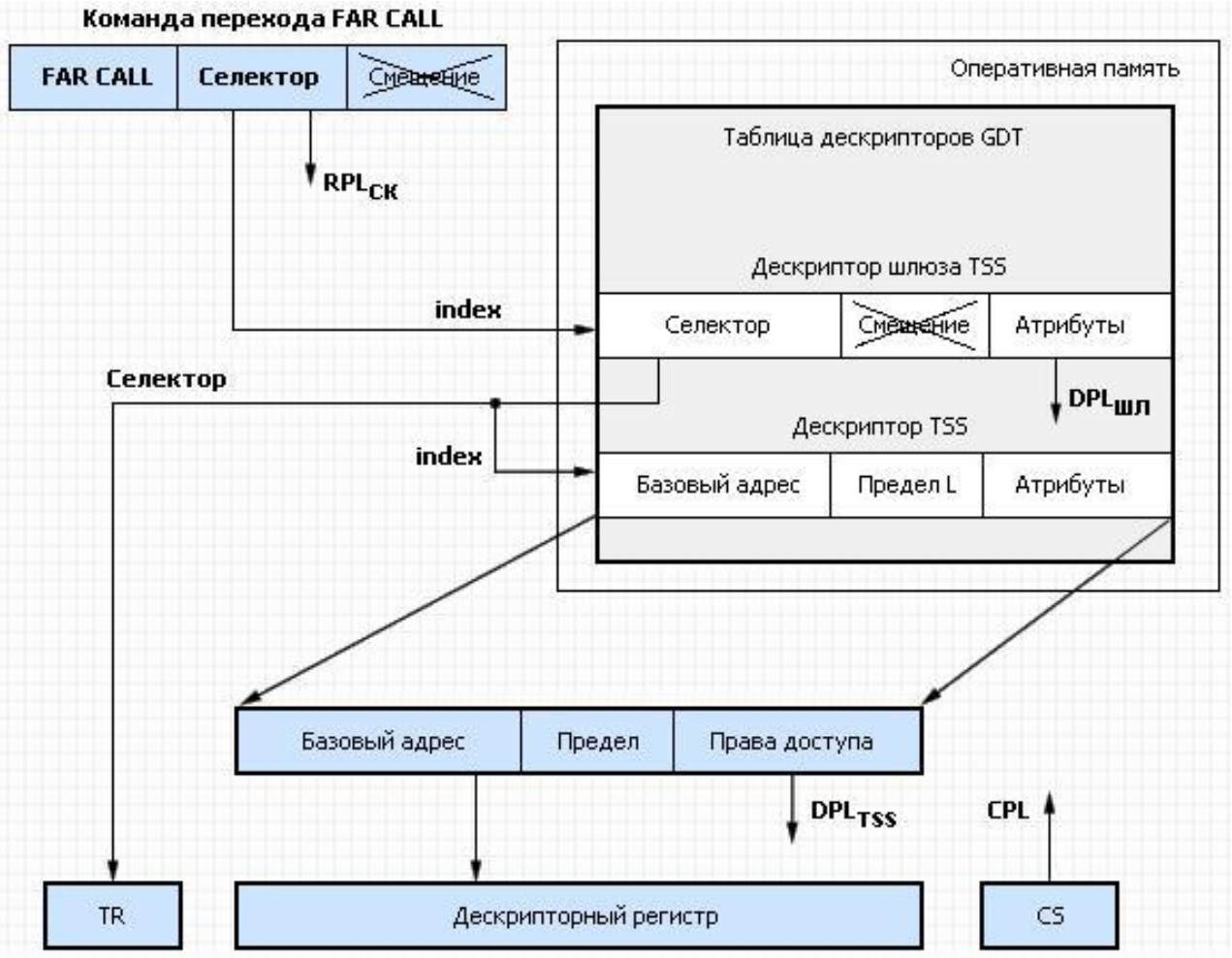
Любое переключение задач на программы более высокого уровня привилегий допускается только при помощи команд CALL типа Far, IRET, INT n, а также в результате прерываний и ловушек. При этом использование шлюза задачи (шлюза TSS) - обязательно.

Со схемой проверок по соответствию уровней привилегий и с процедурой выбора целевого TSS при переключении задач с использованием шлюза задачи по командам прерывания INT n и ловушкам, а также по командам возврата из прерываний IRET можно ознакомиться по схеме или по экранной модели (мод. 8 – см. электронный учебник).

Со схемой проверок по соответствию уровней привилегий и с процедурой выбора целевого TSS при переключении задач с использованием шлюза задачи можно ознакомиться по схеме (рис. 13) или по экранной модели (мод. 9 – см. электронный учебник).

Шлюзы задачи могут храниться как в глобальной, так и в локальных таблицах дескрипторов.

Условия доступа здесь аналогичны условиям доступа при прямых переключениях задач, но вместо сегментов задачи (TSS) используются шлюзы задачи (ШлTSS).



Проверки привилегий по условиям доступа:

1. *($CPL_{пр}$ <= $DPL_{шл}$) & ($RPL_{ск}$ <- $DPL_{шл}$) - Проверка доступности шлюза TSS*
2. *DPL_{TSS} <= CPL - Проверка доступности целевого TSS*

Рис.13. Схема проверок по соответствию уровней привилегий и процедура выбора
целевого TSS при переключении задач с использованием шлюза задачи

Переключение задачи по команде JMP или CALL типа FAR допускается, если уровень привилегий шлюза TSS не выше текущего уровня привилегий и уровня привилегий запроса, т.е.:

$$(CPL <= DPL_{ШлTSS}) \ \& \ (RPL <= DPL_{ШлTSS}),$$

где $DPL_{ШлTSS}$ - уровень привилегий шлюза задачи.

Условия доступа здесь аналогичны условиям доступа при прямых переключениях задач, но вместо сегментов задачи (TSS) используются шлюзы задачи (ШлTSS).

$$(CPL <= DPL_{ШлTSS}) \ \& \ (RPL <= DPL_{ШлTSS}),$$

Проверки соответствия уровней CPL, RPL и DPL_{TSS} не требуется.

Операционная система, создавая шлюзы задач в глобальной или локальных таблицах дескрипторов с низкими уровнями привилегий, может допускать программы нижнего уровня привилегий к разрешенным сервисным процедурам.

При переключении задач с использованием шлюза дескриптор TSS целевой задачи определяется селектором шлюза задачи. При этом, уровни привилегий шлюза TSS и дескриптора TSS могут не совпадать. Следовательно, использование шлюзов TSS допускает любые межуровневые переключения задач.

При использовании команд IRET, команды вызова прерываний, например, INT n или в случаях прерываний и ловушек, как прямое переключение задач, так и переключение задач с использованием шлюзов задач производится вне зависимости от значений, соответственно, DPL_{TSS} или $DPL_{ШлTSS}$, т.е. **без проверок условий доступа по уровням привилегий.**

Но здесь вполне контролируемая ситуация.

Вызовы прерываний, прерывания и ловушки - это всегда переходы или переключения без уменьшения уровней привилегий. Использование шлюза и отсутствие передач параметров является достаточной мерой защиты программ от взаимных помех.

Команда IRET - это команда возвращения из процедуры в исходный кодовый сегмент. Команда не содержит адресного поля. При переключениях задач адрес возврата берется не из стека, а из защищенного системного сегмента TSS, не доступного пользовательским программам. В этой ситуации нет необходимости в проверках допустимости переходов по соотношениям уровней привилегий.

Недостатками использования процедуры переключения задач являются:

- *не поддерживают реентерабельность процедур;*
- *завышенное время переключение программ.*

Кроме проверок по уровням привилегий, процедура переключения задач, особенно с использованием шлюза содержит множество стандартных проверок процедур загрузки селекторов. Переключение задач предполагает перезагрузку всех шести сегментных регистров и селектора локальной таблицы дескрипторов, а также двойное обращение к таблице дескрипторов. Переключение задач с использованием шлюза занимает около 200 тактов работы процессора.

Вопросы для самопроверки:

1. *Основные этапы процедуры переключения задач.*
2. *Допустимые средства переключения задач.*
3. *В чем причина "ослабления" ограничений при использовании переходов с изменением уровней привилегий.*
4. *Как выбирается механизм возврата.*
5. *Как используется бит вложенности NT.*
6. *В чем заключается особенность использования команды JMP для переключения задач.*
7. *В чем суть прямого переключения задач.*
8. *Какие средства допустимы для прямого переключения задач.*

9. Условия допустимости прямых межуровневых переключений задач.

10. Условия допустимости переключения задач с увеличением уровня привилегий (средства, тип обработчика), соотношения уровней привилегий.

11. Условия допустимости переключения задач с увеличением уровня привилегий при использовании команды CALL типа FAR.

12. Условия допустимости переключения задач с увеличением уровня привилегий при использовании команды INT n, прерывания или ловушки.

2.4.3.6. Обобщенная модель механизма защиты информации.

Механизм защиты по уровням привилегий реализует множество проверок корректности загрузки сегментов данных и межсегментных переходов, включая возвраты из вызываемых процедур (разд. 2.4.2 и 2.4.3).

Процедуры проверок являются функциями:

1. Для загрузки сегментов данных - типа загружаемого селектора:

- *DS, ES, FS, GS*
- *SS*

2. Для переходов:

- *типа целевого сегмента перехода (подчиненного, неподчиненного);*
- *выбранного обработчика механизма перехода (передачу управления, переключение задач),*
- *средств перехода (команду JMP, CALL типа FAR, RET, RET n, команду вызова прерываний, прерывания, ловушки).*

Комбинации указанных аргументов определяют условия корректности:

- *межуровневого доступа к данным;*
- *межуровневых переходов с изменением уровня привилегий.*

Обобщенная модель механизма защиты информации является результатом обобщения и систематизации всех возможных правил определения корректности доступа к данным и переходов по соотношению уровней привилегий.

Для описания функционирования механизма защиты информации МП Intel можно воспользоваться моделью в виде двумерной таблицы (Таблица).

В модели (Таблица) все возможные правила определения корректности доступа к данным сведены к десяти (два - для обмена данными, восемь - для переходов).

Условные сокращения

В условиях доступа по уровням привилегиям в Таблице используются сокращения:

- *CPL - текущий уровень привилегий (из селектора CS);*
- *RPL - уровень привилегий запроса (из селектора команды);*
- *DPL - уровень привилегий целевого кодового сегмента (из дескриптора целевого кодового сегмента);*
- *$DPL_{шл}$ - уровень привилегий шлюза (из дескриптора используемого шлюза);*
- *DPL_{TSS} - уровень привилегий TSS (из дескриптора используемого TSS);*
- *$DPL_{ШлTSS}$. - уровень привилегий шлюза TSS (из дескриптора используе-*

мого шлюза TSS).

Сводная таблица проверок доступа по соответствию уровней привилегий.

<table>
<tr><td colspan="5">Загрузка сегментных регистров данных</td></tr>
<tr><td colspan="4">Средство загрузки</td><td>Условие доступа по привилегиям</td></tr>
<tr><td colspan="4">Команды:(LDS, LES, LFS, LGS, LSS) reg, mem</td><td>(CPL <= DPL) & (RPL <= DPL)</td></tr>
<tr><td colspan="4">Команда LSS reg, mem</td><td>CPL = DPL = RPL</td></tr>
<tr><td colspan="5">Взаимодействие программ</td></tr>
<tr><td>Механизм перехода</td><td>Обработчик</td><td>Средство перехода</td><td>Условия доступа по привилегиям</td><td>Изменение уровня приоритета</td></tr>
<tr><td rowspan="4">Передача управления</td><td rowspan="2">кодовый сегмент</td><td>FAR JMP, FAR CALL (на подчиненные сегменты)</td><td>DPL <= CPL</td><td rowspan="2">без изменения</td></tr>
<tr><td>FAR JMP (на неподчиненные сегменты)</td><td>(CPL = DPL) & (RPL <= CPL)</td></tr>
<tr><td rowspan="2">шлюзы: вызова, прерывания или ловушки</td><td>по команде FAR CALL</td><td>(CPL <= $DPL_{шл}$) & (RPL <= $DPL_{шл}$) & (DPL = CPL)</td><td rowspan="2">без уменьшения</td></tr>
<tr><td>по командам прерывания, прерываниям и ловушкам</td><td>(CPL <= $DPL_{шл}$) & (DPL <= CPL)</td></tr>
<tr><td>кодовый сегмент</td><td>RET, RET n и IRET</td><td>$CPL_{в}$ = $DPL_{в}$ <= CPL</td><td>без увеличения</td></tr>
<tr><td rowspan="3">Переключение задач</td><td>TSS</td><td>по командам: FAR JMP, FAR CALL</td><td>(CPL <= DPL_{TSS}) & (RPL <= DPL_{TSS})</td><td>без увеличения*</td></tr>
<tr><td>TSS или шлюз TSS</td><td>по командам прерывания и IRET, по прерываниям и ловушкам</td><td>без условий</td><td rowspan="2">любые</td></tr>
<tr><td>шлюз TSS</td><td>по командам: FAR JMP FAR CALL</td><td>(CPL <= $DPL_{шлTSS}$) & (RPL <= $DPL_{шлTSS}$)</td></tr>
</table>

2.5. ГЛОССАРИЙ.

Адресное пространство программы - совокупность логических адресов, занимаемая программой.

Байт прав доступа (AR - байт в составе поля атрибутов дескрипторов сегментов). AR содержит поля:

- P (Present) - бит присутствия сегмента в оперативной памяти.
- DPL (Descriptor Privilege Level) - двухразрядное поле, определяющее уровень привилегий сегмента.
- S - тип дескриптора сегмента. При S = 0 - дескриптор определяет сегменты кодов или данных (включая стеки). При S = 1 - системный дескриптор.
- TIPE - см. поле типов.

Дескриптор - совокупность бит, определяющая основные программные объекты: сегменты и шлюзы.

Дескриптор сегмента - дескриптор, определяющий сегмент как совокупность смежных адресов, начиная с нулевого адреса. Содержит следующие поля:

- Поле базового адреса, определяющее расположение сегмента в физической памяти.
- Поле предела (L - Limit), определяющее размер сегмента.
- Поле атрибутов, определяющее тип сегмента и права доступа.

Дескриптор сегмента состояния задач - дескриптор системного сегмента TSS (см. сегмент состояния задачи).

Дескриптор шлюза - определяет разрешенную точку входа в программу. В зависимости от средства и механизма перехода на программу используются шлюзы вызова, ловушки, прерывания и задачи. Дескриптор шлюза содержит следующие поля:

- Поле селектора сегмента.
- Поле смещения в сегменте (в шлюзе задачи не используется).
- Поле атрибутов.

Динамическая переадресация программ - преобразование (пересчет) адресов операндов и переходов в процессе выполнения каждой команды с учетом расположения программы в оперативной памяти.

Защищенный режим работы - работа (для МП intel) в 32-разрядном режиме (IA32).

Команды обращения к устройствам ввода/вывода - команды подгруппы IOPL-"чувствительных" команд. На их выполнение, кроме соотношения уровней привилегий CPL<= IOPL, влияет содержимое битовой карты разрешения ввода/вывода в сегменте состояния задачи TSS.

Команды, модифицируемые в соответствии с текущим уровнем привилегии - непривилегированные команды, выполнение которых зависит от значения CPL. Это две команды: POPFD и POPF (загрузка регистра EFLAGS соответственно четырехбайтным или двухбайтным значением из стека). Эти команды могут изменять любые биты флагов, но биты IOPL могут быть изменены этими командами, только если выполняется PL0-программа, а флаг IF - если выполняется условие CPL <=IOPL

Логический адрес - адрес, заданный двумя компонентами: селектором, определяющим расположение сегмента в физической памяти, и смещением в сегменте.

Мандатные поля строк каталога и таблиц страниц - поля строк каталога или таблиц механизма виртуальной памяти. Используются для дополнительной защиты при преобразовании линейных адресов в физические.

Нуль-селектор - селектор с нулевым значением индекса. Определяет "пустой" дескриптор.

Передача управления - тип программного перехода. Осуществляется по командам переходов или по прерываниям путем загрузки нового адреса в указатель команд (IP). В командах переходов с возвратом - осуществляется запоминание логического адреса возврата в стеке, при прерываниях в стеке дополнительно запоминается содержимое регистра флагов (EFLAGS).

Переключение задач (смена контекста) - тип программного перехода. Осуществляется механизмом, который сохраняет состояния основных регистров процессора в сегменте состояния текущей задачи (TSS), а затем загружает регистры процессора новыми значениями из сегмента состояния задачи (TSS) программы - цели.

Подчиненные кодовые сегменты - кодовые сегменты, отмеченные как подчиненные битом C (Conforming) в поле атрибутов дескриптора сегмента. Передача управления на подчиненный сегмент разрешена, если его уровень привилегий не ниже текущего. Передача управления на подчиненные сегменты не изменяет текущий уровень привилегий.

Поле атрибутов - поле дескриптора, определяющее тип программного объекта и права доступа.

Поле индекса - поле селектора, определяющее вход в таблицу дескрипторов.

Поле предела (L- limit) - двадцатиразрядное поле дескриптора сегмента, определяющее размер сегмента. Задается в байтах или в страницах по 4096 байт в зависимости от бита гранулярности G (Granularity). При G = 0 сегмент задается в байтах, а при G = 1 - в страницах.

Поле уровня привилегии источника запроса RPL (Requested Privilege Level) - уровень привилегий, заданный селектором команды загрузки сегментных регистров или селектором команды межсегментной передач управления.

Поле типов сегментов TIPE - четырехразрядное поле, интерпретируемое по-разному в зависимости от типа дескриптора. В системных дескрипторах (S = 1) разряды поля TIPE кодируют подтипы системных дескрипторов (см. системные дескрипторы). В несистемных дескрипторах (S = 0) определяют: тип сегментов (данные/команды бит - E), направление расширения для сегмента данных или подчиненность для кодового сегмента (бит – ED/C), права использования (бит – W/R) и факт обращение к сегменту (бит – A).

Реальный режим работ - работа в режиме МП i8086.

Сегмент состояния задачи TSS - системный сегмент, предназначенный для сохранения содержимого основных регистров процессора при переключении задач.

Смена контекста - см. переключение задач.

Текущий уровень привилегии CPL (Current Privilege Level или Code Privilege Level) - уровень привилегий выполняемого кодового сегмента. При проверках берется из поля уровня привилегий селектора кодового сегмента.

Уровень привилегий ввода/вывода IOPL - двухразрядное поле регистра флагов (EFLAGS), определяющее возможности использования IOPL - "чувствительных" команд. Это команды, которые изменяют состояние флажка прерываний IF, выполняют захват шины или операцию ввода-вывода. Для выполнения этих команд программа необязательно должна иметь уровень привилегий 0. Привилегированность этих команд заключается в том, что их могут выполнять программы, уровень привилегий которых выше уровня, определяемого полем IOPL уровня привилегий ввода-вывода в регистре EFLAGS.

Уровень привилегий дескриптора DPL (Descriptor Privilege Level) - уровень привилегий программного объекта (сегмента или шлюза). Задается полем уровня привилегий соответствующего дескриптора.

CPL - см. текущий уровень привилегий.

DPL - см. уровень привилегий дескриптора.

$DPL_{цкс}$ - уровень привилегий целевого кодового сегмента.

$DPL_{шл}$ - уровень привилегий шлюза.

IOPL -"чувствительные" (IOPL-sensitive) команды - см. Уровень привилегий ввода/вывода IOPL

L - см. поле предела.

PL0-команды - привилегированные команды, использование которых допускается только в кодовых сегментах с нулевым уровнем привилегий.

TIPE - см. поле типа сегмента.

TSS - см. сегмент состояния задачи.

Теоретический материал также приведён в электронной обучающей программе.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Лабораторная работа расположена по следующему пути на сервере кафедры:
Z:\Документация\По предметно\Организация ЭВМ и систем (ОЭВМиС) \Лабораторные занятия\ Лабораторная работа №6\protection posl

2. Скопируйте папку «**protection posl**» на свой компьютер.

3. Внутри папки запустите файл **protection.exe.**

4. Изучите теоретический материал, представленный в программе.

4. СОДЕРЖАНИЕ ОТЧЕТА

Отчет должен включать устные ответы на вопросы преподавателя.