# Архитектура вычислительных систем и компьютерных сетей

Вощинская Гильда Эдгаровна

# *Защита памяти*

# *Средства защиты памяти должны предотвращать*

- неразрешенное взаимодействие пользователей друг с другом,
- несанкционированный доступ пользователей к данным,
- повреждение программ и данных из-за ошибок в программах,
- намеренные попытки разрушить *целостность системы*,
- использование информации в памяти не в соответствии с ее функциональным назначением.

# *Методы защиты базируются*

на некоторых классических подходах, которые получили свое развитие в архитектуре современных ЭВМ.

К таким методам можно отнести

- защиту отдельных ячеек,
- *метод граничных регистров*,
- *метод ключей защиты.*

# *Средства защиты памяти в персональной ЭВМ*

*Защита памяти* в *персональной ЭВМ* делится

- на *защиту при управлении памятью и*
- *защиту по привилегиям*.

# *Средства защиты при управлении памятью*

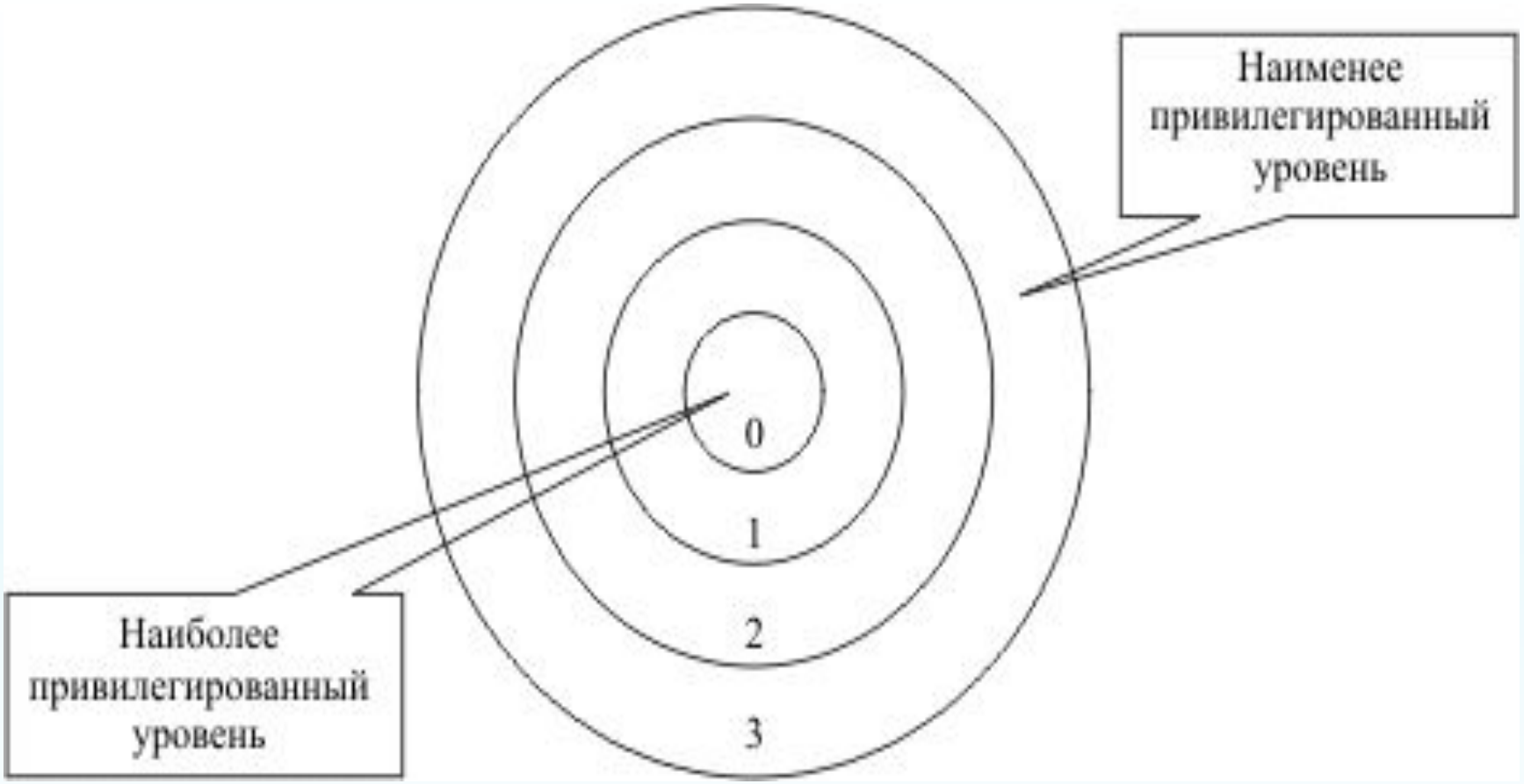
- осуществляют проверку превышения *эффективным адресом* длины сегмента;
- прав доступа к сегменту на запись или только на чтение;
- функционального назначения сегмента.

# *Защита по привилегиям*

Различным объектам (программам, сегментам памяти, запросам на обращение к памяти и к внешним устройствам), которые должны быть распознаны процессором, присваивается идентификатор, называемый *уровнем привилегий* . *Процессор* постоянно контролирует, имеет ли текущая программа достаточные привилегии, чтобы

- выполнять некоторые команды,
- выполнять команды ввода-вывода на том или ином внешнем устройстве,
- обращаться к данным других программ,
- вызывать другие программы

# Кольца защиты



# *Распределение программ по кольцам защиты*

*уровень 0* - *ядро* ОС, обеспечивающее инициализацию работы, *управление доступом* к памяти, защиту и ряд других жизненно важных функций, нарушение которых полностью выводит из строя *процессор*;

*уровень 1* - основная часть программ ОС (утилиты);

*уровень 2* - служебные программы ОС (драйверы, *СУБД*, специализированные подсистемы программирования и др.);

# *Уровни привилегий находятся в 3-х различных местах*

Каждый дескриптор имеет *уровень привилегий дескриптора DPL.*

Каждый селектор содержит *запрашиваемый уровень привилегий RPL*.

Процессор 80286 поддерживает *текущий уровень привилегий CPL*. Он показывает уровень благонадежности выполняющейся программы и равен содержимому поля DPL сегмента, который адресует регистр CS в двух младших битах CS, замещая поле RPL хранимого там селектора.

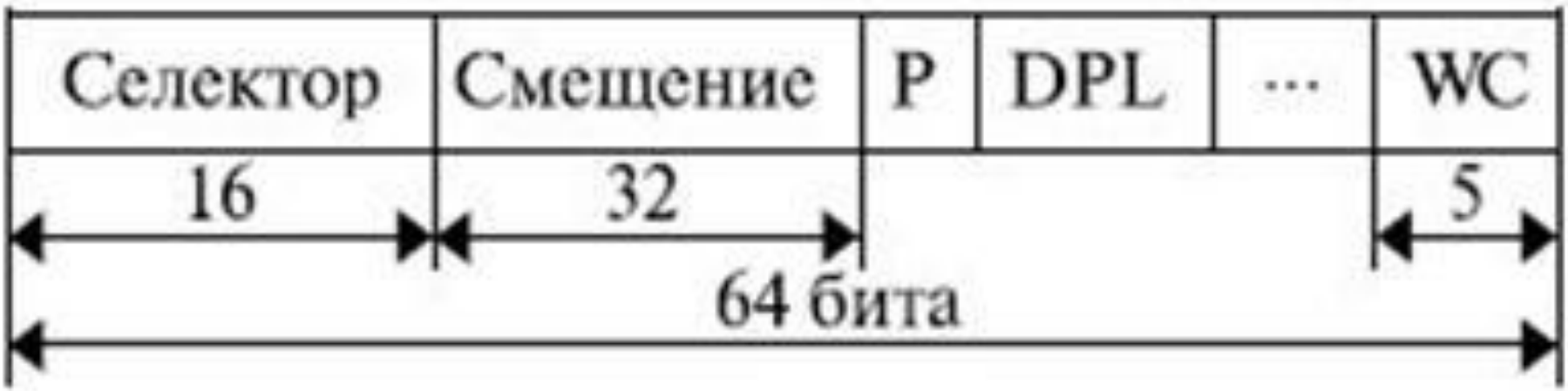
# *Доступ из одного сегмента к другому*

разрешается, если DPL>=CPL( чтобы программа не могла изменить поле RPL правило приобретает вид: DPL>=max(CPL,RPL)).

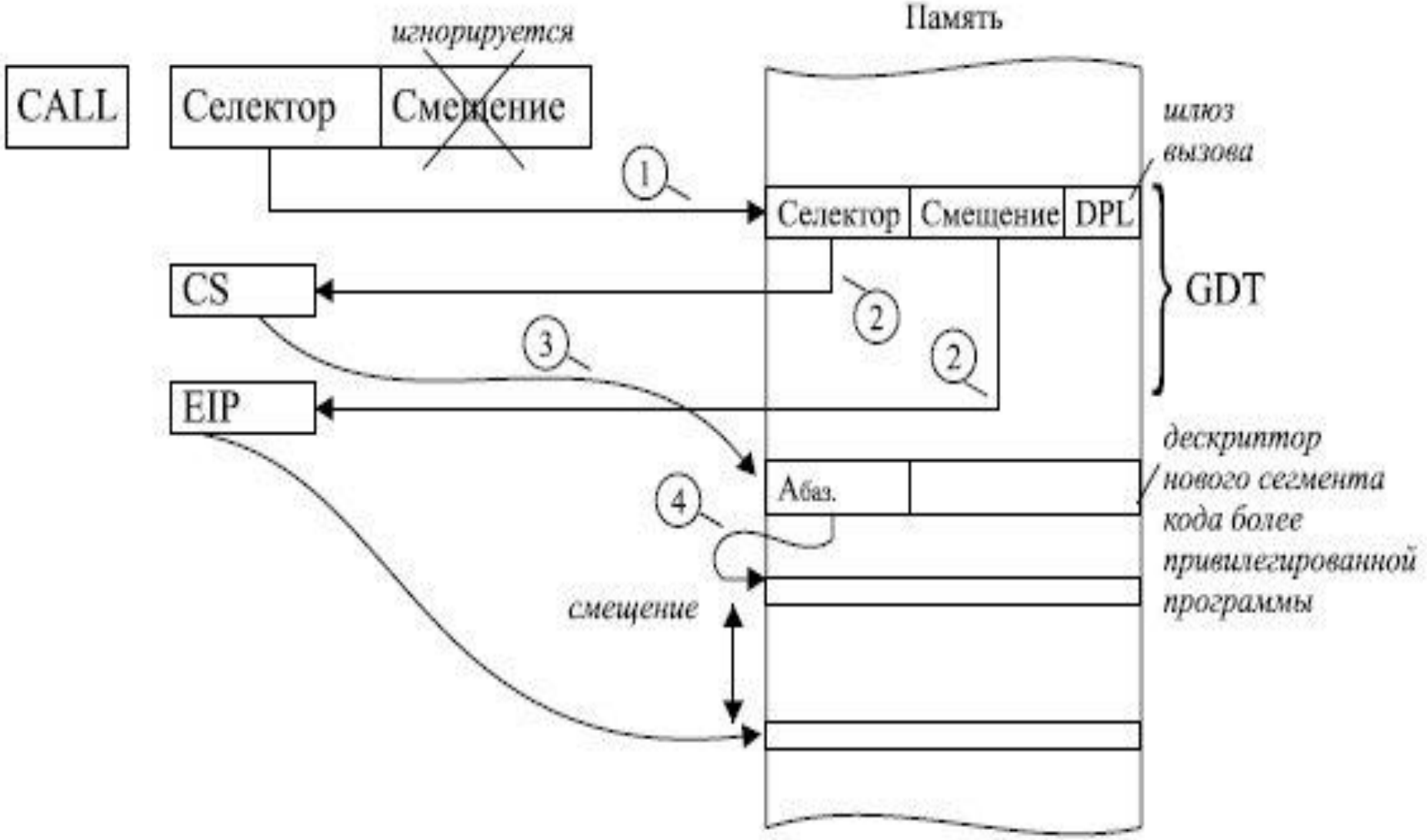
Если к какой-либо системной программе предусматривается обращение со стороны менее привилегированных программ, то для нее создается специальный объект – *шлюз вызова*.

# *Шлюз вызова*

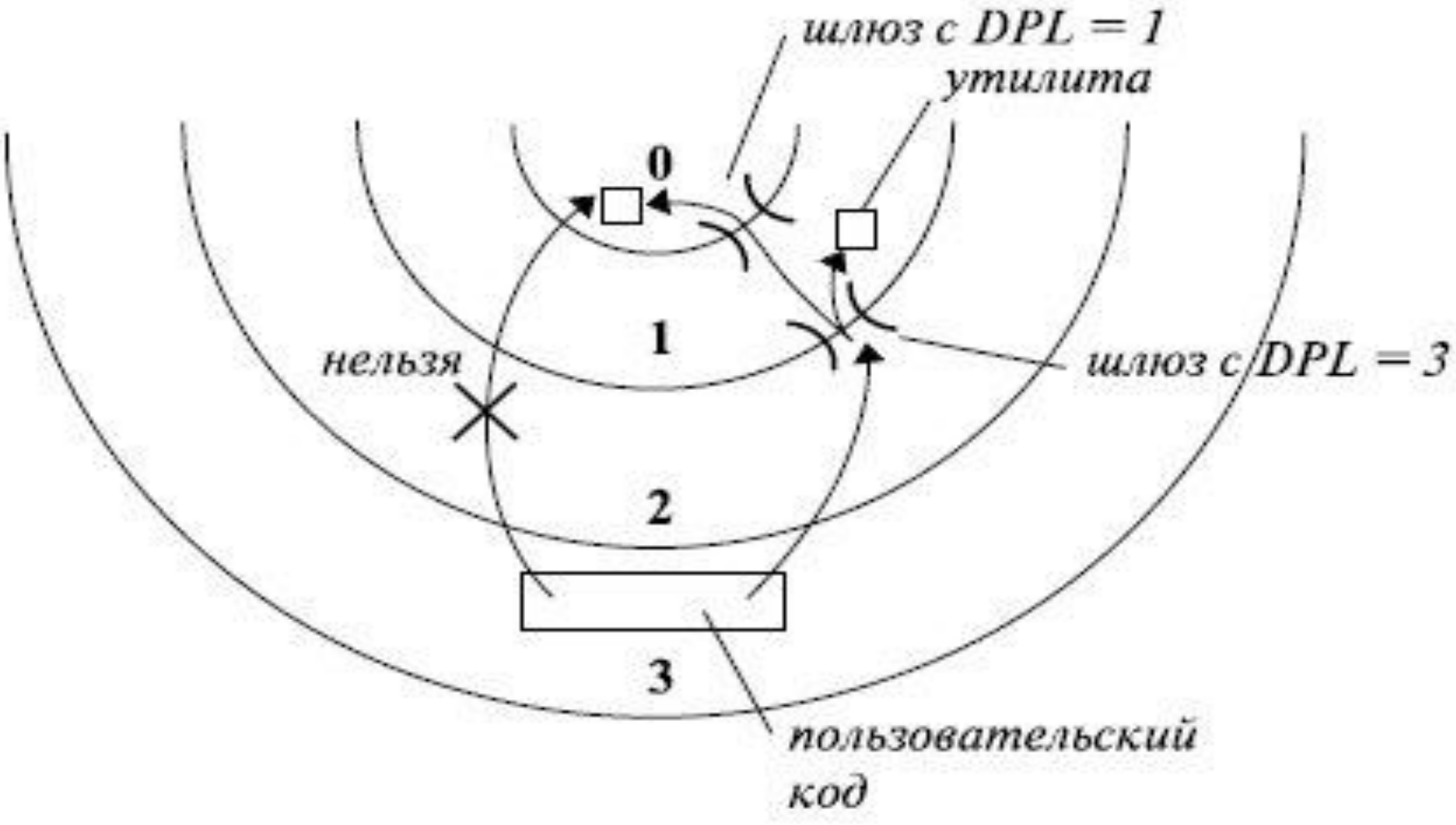
- специальный дескриптор, определяющий точку входа в соответствующие процедуры.



# Использование шлюза вызова для обращения к программам на более высоком уровне привилегий



# Последовательное обращение к более привилегированным программам



Чтобы проанализировать механизм вызовов и возвратов в виртуальном режиме, рассмотрим три случая:

1) вызывающая и вызываемая программы имеют одинаковые привилегии;
2) вызывающая программа менее привилегированна, чем вызываемая;
3) вызывающая программа более привилегированна, чем вызываемая.

# *Если вызывающий и вызываемый сегменты имеют одинаковые привилегии*

вызовы и возвраты действуют так же, как в реальном режиме. Это остается справедливым, даже если вызов проходит через шлюз: он просто перенаправляет назначение вызова.

# *Если вызывающий сегмент менее привилегирован, чем вызываемый*

вызов должен проходить через шлюз.

Кроме того, вызываемая процедура должна использовать другой, более привилегированный сегмент стека, чем вызывающая процедура.

# *Если вызывающий сегмент менее привилегирован, чем вызываемый*

Для использования своего стека привилегированной программой есть две причины.

1. Вызываемая процедура должна защищаться от возможного переполнения стека, которое возникает, если вызывающая процедура распределяет недостаточное стековое пространство.
2. Возможно, что менее привилегированный сегмент стека менее привилегированной вызывающей процедуры разделяется с

# *Порядок передачи параметров*

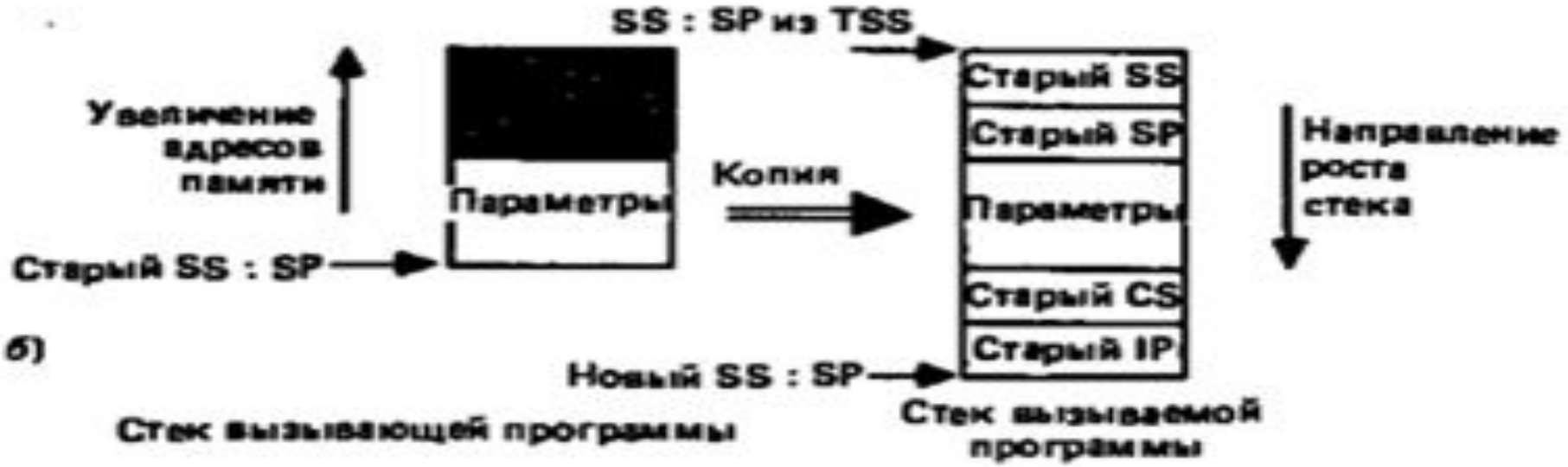
Процессор 80286 получает новые значения регистров SS и SP для нового стека из специального сегмента, называемого сегментом состояния задачи TSS и ассоциируемого с задачей. Каждый TSS содержит три потенциальных пары значений для регистров SS и SP: по одной для каждого из колец О, 1 и 2. Пара для кольца 3 не нужна, так как не существует вызова, в котором вызывающая процедура

# *Порядок передачи параметров*

После загрузки регистров SS и SP для адресации нового стека процессор включает в него старые значения из регистров SS и SP. Затем он копирует все параметры из старого стека в новый. Поле счетчика из шлюза вызова определяет, сколько слов копировать. Наконец, процессор включает в новый стек старые значения регистровCS и IP, т. е. адрес возврата.

# Порядок передачи параметров


а)




б)

Содержимое стека после длинного вызова:
а — внутри кольца, б — между кольцами

# *Возврат из межкольцевого вызова*

При возврате из межкольцевого вызова команда возврата должна произвести следующие действия:

1. Извлечь адрес возврата и поместить его в регистры CS и IP.
2. Извлечь параметры из стека вызываемой программы.
3. Переключиться на стек вызывающей программы, извлекая старые значения SS и SP из стека и помещая в регистры SS и SP.

# *Возврат из межкольцевого вызова*

4. Извлечь параметры из стека вызывающей программы.
5. Продолжить выполнение вызывающей программы.

Команда возврата сбросит содержимое регистров DS или ES, если любой из них адресует сегмент, более привилегированный, чем вызывавшая программа. Это действие запрещает вызывающей программе недопустимый доступ к привилегированным сегментам, случайно оставленным

# *Вызывающая программа более привилегированна, чем вызываемая*

В этом маловероятном случае ОС запрашивает обслуживание от пользовательской программы, что фактически запрещается процессором 80286.

Причина этого, в том, что команда возврата передает управление из менее привилегированного кольца в более привилегированное. Подобные вызовы и возвраты приводят к особому случаю

# *Подчиненные сегменты*

В дескрипторах исполняемых сегментов (и только для таких сегментов) имеется бит *подчинения C*. Если этот бит установлен в 1, то сегмент называется *подчиненным*. Процедуры в подчиненном сегменте можно вызывать или переходить к ним прямо из программы, не более привилегированной, чем подчиненный сегмент: поле CPL вызывающей программы должно быть больше или равно полю DPL подчиненного сегмента. Шлюз вызова не требуется.

# *Подчиненные сегменты*

Стеки не переключаются. Когда производится передача управления подчиненному сегменту, поле CPL не изменяется; оно остается на уровне вызывающей программы. При вызове подчиненный сегмент выполняется с привилегиями вызвавшей его программы. Следовательно, подчиненный сегмент автоматически "перемещается" или "подчиняется" кольцу вызывающей программы.

# *Мультизадачность*

Процессор 80286 поддерживает несколько задач путем переключения с одной задачи на другую. Для этого он ассоциирует с каждой задачей сегмент памяти, содержащий всю информацию, необходимую для запуска и останова задачи. Этот специальный сегмент называется *сегментом состояния задачи TSS.* Кроме того, в процессоре имеется 16-битный регистр задачи, содержащий селектор GDT для сегмента TSS текущей выполняющейся задачи.

# *Переключение задач*

Основное назначение сегмента состояния задачи — сохранять содержимое регистров задачи, когда процессор ее не выполняет.

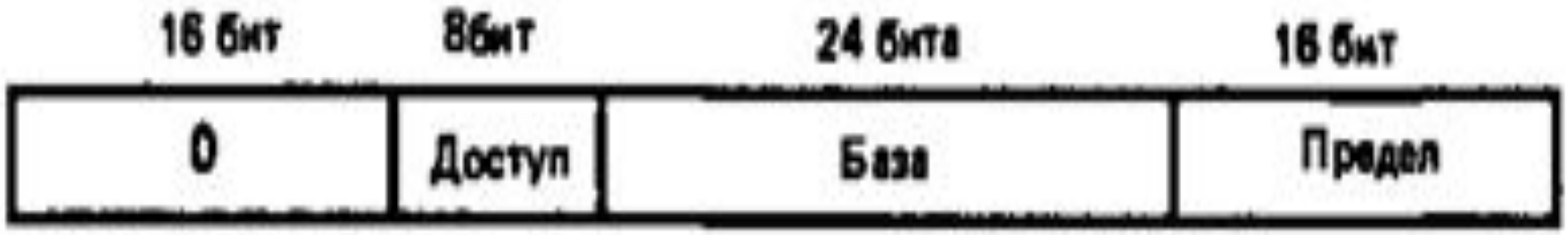
Действия процессора 80286 при переключении на задач :

1) сохраняет все «видимые» программе регистры (за исключением регистра GDT) в TSS, на который показывает регистр задачи;
2) загружает в регистр задачи селектор для нового TSS;
3) загружает свои регистры из нового TSS и

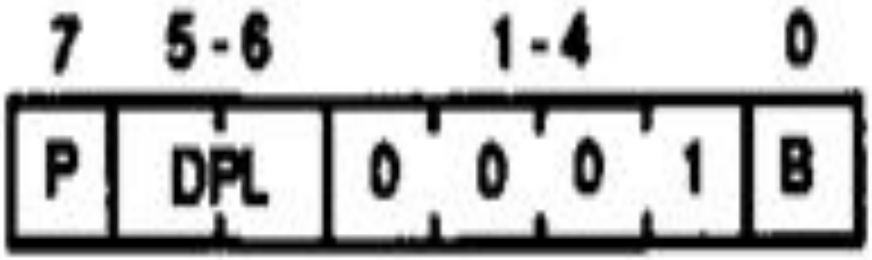
# *Переключение задач*

TSS, как и сегменты LDT, имеют специальные дескрипторы. В регистр задачи можно загружать только селекторы дескрипторов TSS. В дескрипторе TSS в байте доступа бит занятости B используется чтобы предотвратить выполнение одной и той же задачи двумя процессорами одновременно. Процессор 80286 устанавливает в 1 бит занятости в дескрипторе TSS для текущей выполняющейся задачи и сбрасывает его в 0, когда переключается на другую задачу.

# Формат дескриптора TSS




а)

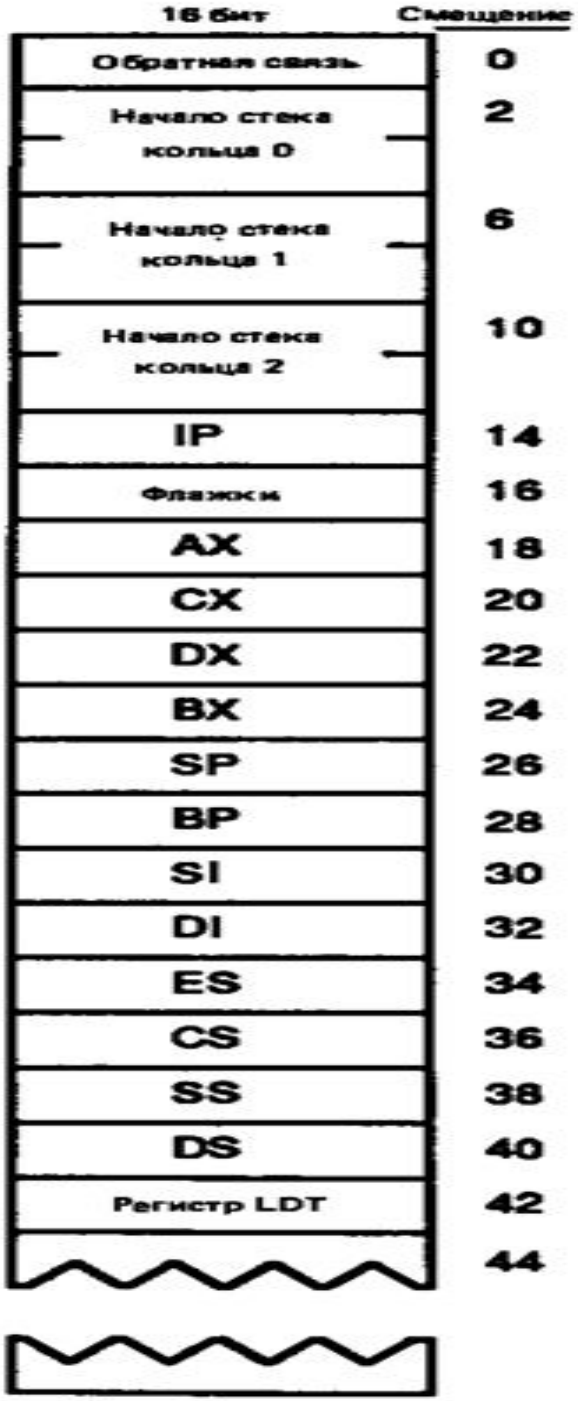



б)

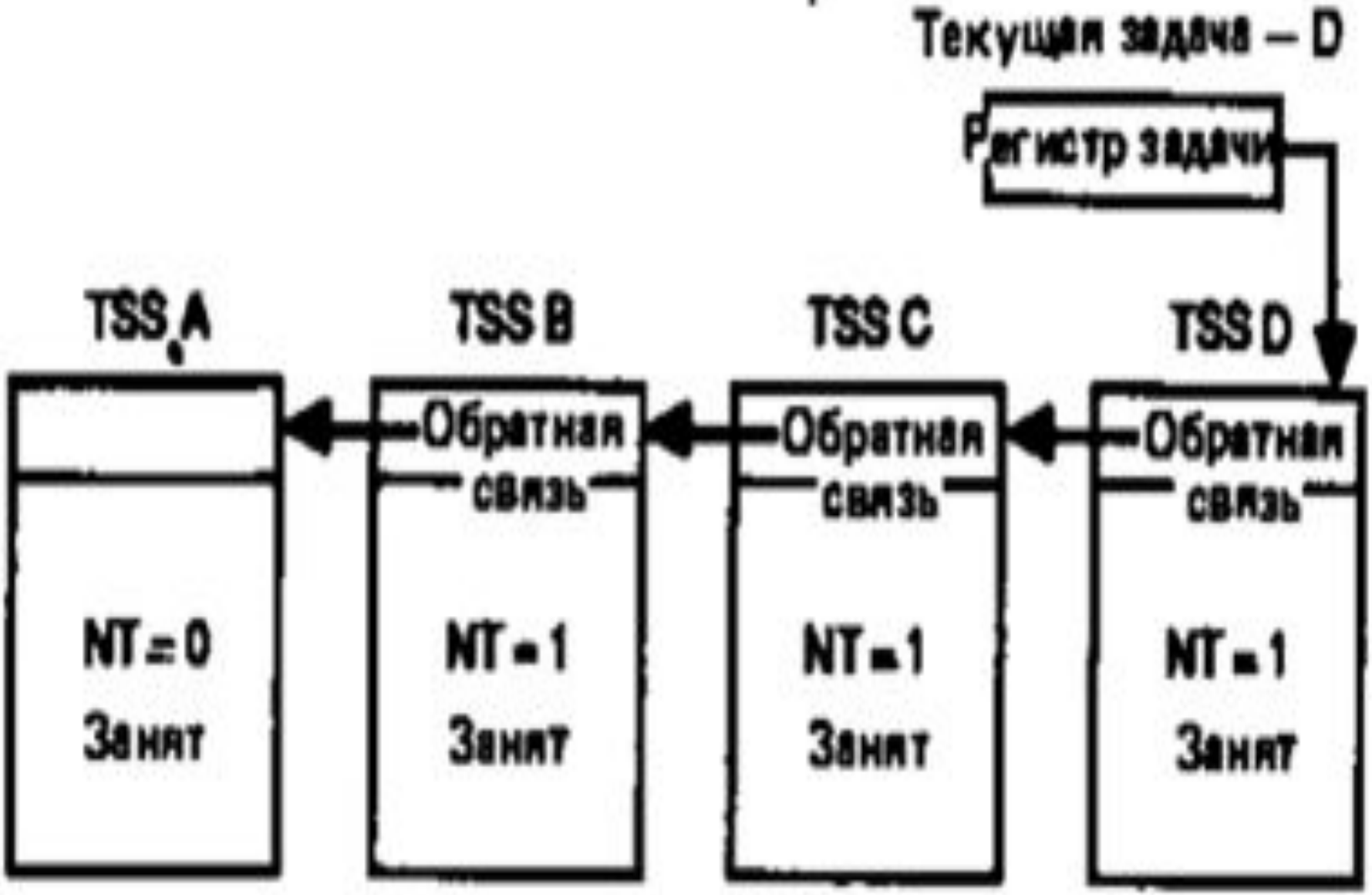
Формат дескриптора сегмента состояния задачи:

а — дескриптор состояния задачи, б — его байт доступа

Формат первых
44 байт TSS
(аппаратно-определенный)



# Пример разделения времени



# *Приоритеты прерываний*

Прерывания от некоторых периферийных устройств являются более срочными или важными, чем от другие.

Требуется упорядочить прерывания в системе, назначив важным из них более высокий приоритет. В самом процессоре 80286 схем для этого нет, поэтому обычно все прерывающие устройства подключаются к микросхеме 8259A контроллера прерываний, а она подключается к процессору 80286.

# *Контроллер прерываний 8259A*

Контроллер прерываний узнает о том, что ЦП закончил обработку прерывания, с помощью специальных команд, которые вставляются в конце процедуры прерывания каждого внешнего устройства; эти команды выводят особый код конца прерывания в порт, подключенный к контроллеру прерываний.

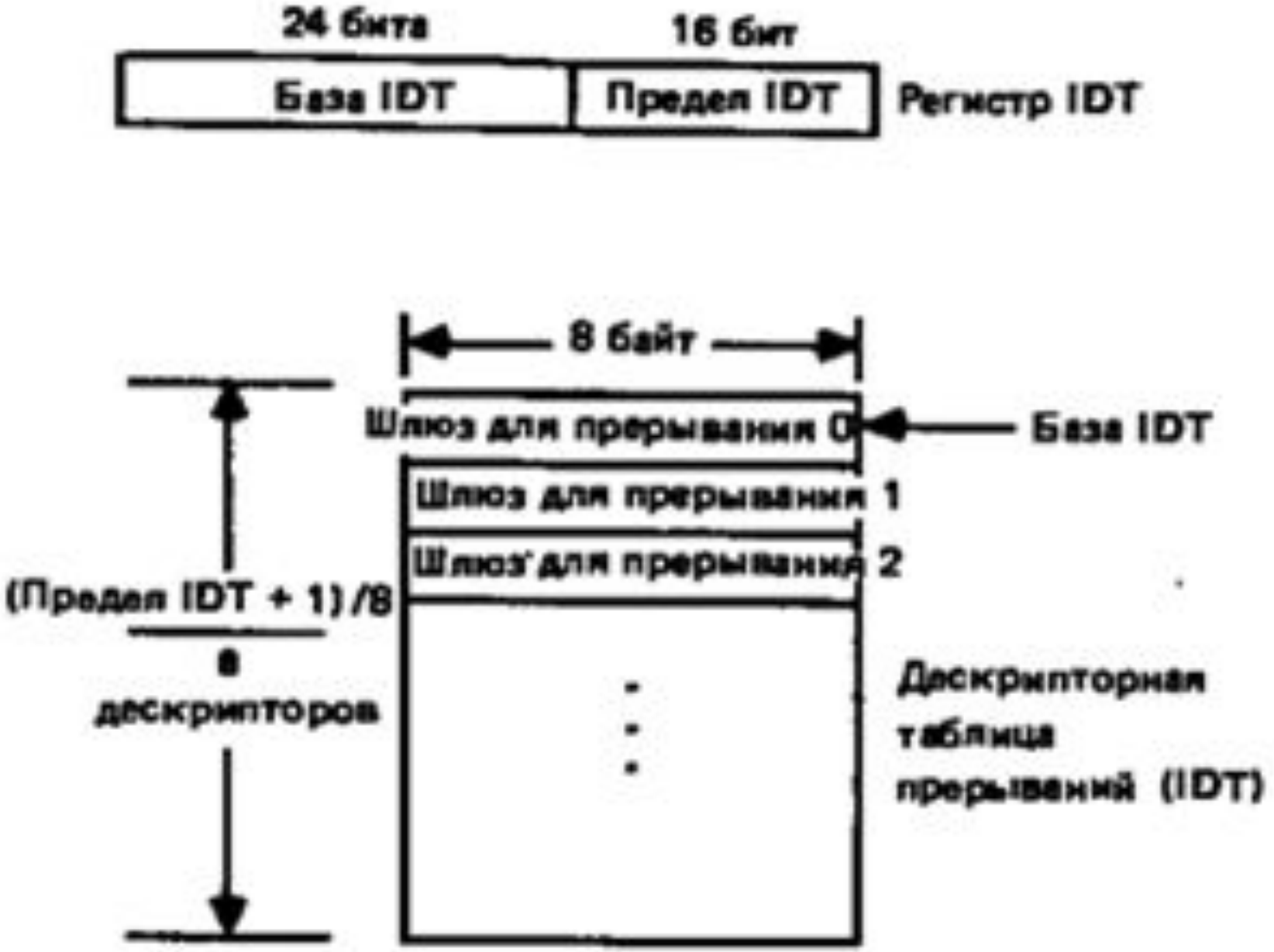
```
MOV DX, inportum
MOV AL, eoicode
OUT DX, AL
```

# *Прерывания в виртуальном режиме*

В виртуальном режиме таблица прерываний содержит дескрипторы различных обработчиков прерываний и называется *дескрипторной таблицей прерываний IDT*.

Базовый адрес и предел таблицы IDT хранятся в регистре IDT.

# Дескрипторная таблица прерываний



# *Шлюзы в таблице IDT*

Шлюзы специальных прерываний — это стандартный вид шлюзов для особых случаев. Процедуры обработки особых случаев, адресуемые шлюзами специальных прерываний, вызываются как в реальном режиме, но при входе в них флажок разрешения прерываний IF остается неизменным, чтобы особые случаи в задаче не выключали механизм разделения времени операционной системы путем игнорирования прерываний таймера.

# *Шлюзы в таблице IDT*

Шлюзы задач — это стандартный вид шлюзов для внешних прерываний. Обработчики прерываний, адресуемые шлюзами задач, инициируются операцией вызова задачи, так как обычно такие сигналы прерываний направляются не в текущую выполняющуюся программу, а в операционную систему.

# *Обработка прерывания в зависимости от задачи*

В некоторых языках высокого уровня, например, программистам разрешено определять собственные обработчики особых случаев. В системах, рассчитанных на такие языки, различные задачи могут потребовать разных обработчиков одного и того же особого случая. Поскольку имеется всего одна IDT, которую разделяют все задачи в системе, придется искать какие-то решения.

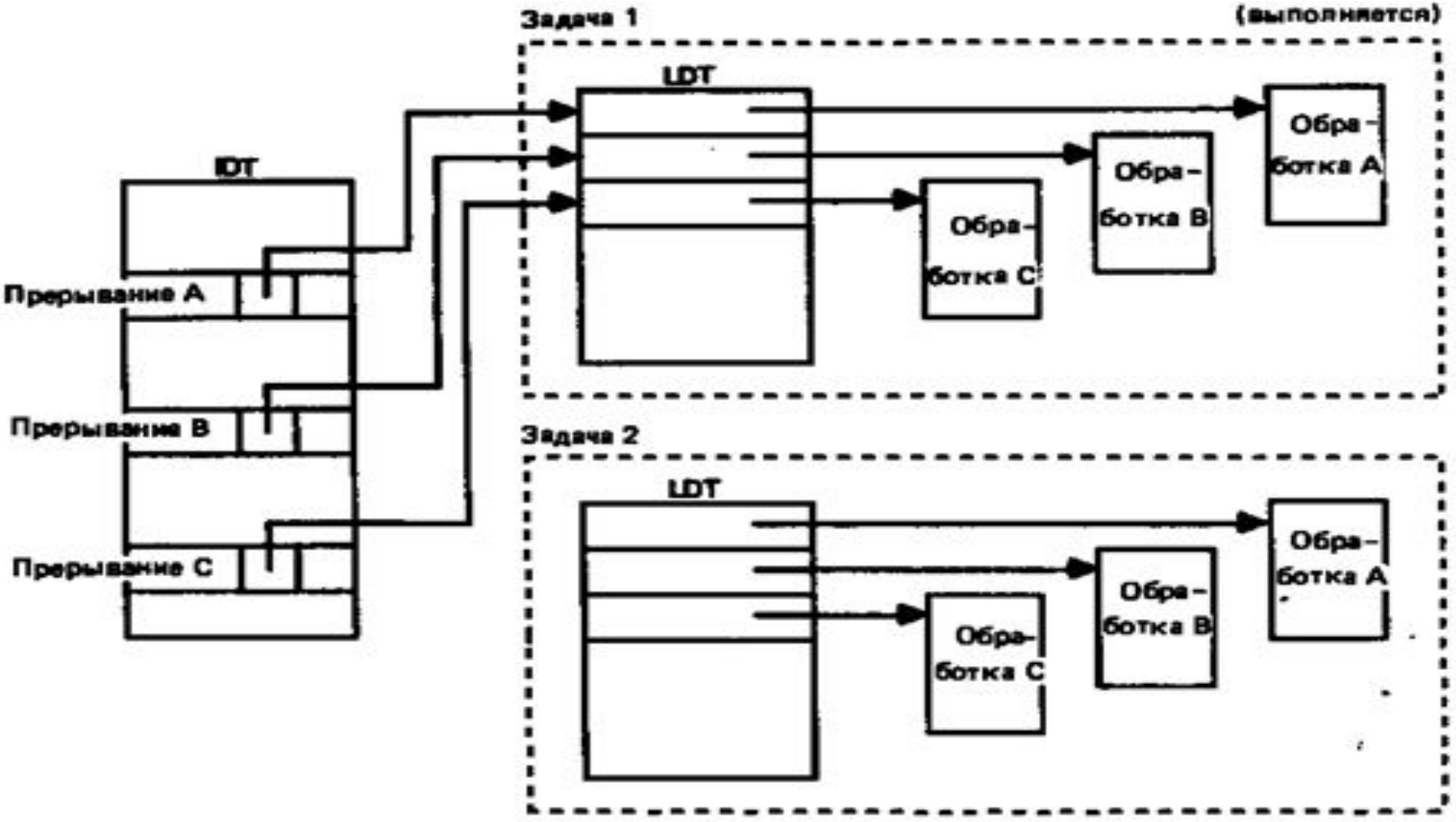
# *Обработка прерывания в зависимости от задачи*

Одно из возможных решений заключается в том, чтобы ввести в операционную систему процедуру распределения прерываний. Все особые случаи, которые ОС разрешает пользователям обрабатывать самим, проходят через распределитель: все шлюзы специальных прерываний для этих особых случаев в таблице IDT адресуют распределитель прерываний.

# *Распределитель прерываний*

Распределитель обращается к таблице, находящейся в локальном адресном пространстве пользовательской программы, и определяет, куда переходить для каждого из особых случаев, поскольку обращение к таблице осуществляется через IDT, она будет меняться в соответствии с задачей, из которой вызвана процедура распределителя прерываний. Сам распределитель прерываний должен находиться в подчиненном сегменте.

# Еще одно решение — превратить в распределитель прерываний сам процессор 80286



# *Коды ошибок*

В виртуальном режиме многие особые случаи обеспечивают обработчику дополнительную информацию, которая идентифицирует характер ошибки. Для таких особых случаев процессор 80286 включает в стек адрес возврата и 16-битный код ошибки, а затем переходит к обработчику особого случая. При использовании шлюза задачи адреса возврата нет, поэтому в стек включается только код ошибки.

# Шестиуровневый компьютер