



# **Аппаратная поддержка многозадачности. Часть 3. Система прерываний.**

Лектор: доц., к.т.н.,  
доцент кафедры ИСТАС  
Иванов Н.А.

2019 г.



Современная ЭВМ представляет собой комплекс автономных устройств, каждое из которых выполняет свои функции под управлением местного устройства управления независимо от других устройств машины.

Устройство включается в работу центральным процессором (ЦП), который передает устройству команду и все необходимые для ее исполнения параметры.

После начала работы устройства центральный процессор отключается от него и переходит к обслуживанию других устройств или к выполнению других функций.



Во время работы вычислительной системы к ЦП поступает от других устройств или вырабатывается в нем самом большое количество различных сигналов.

Множество сигналов, которые ЦП в состоянии воспринять, учесть и обработать, **определяется выполняемой** процессором **программой**.

Назовем это множество *зоной внимания* ЦП.



Пусть «**зону внимания**» программы кроме 2-х регистров данных составляет еще и регистр флагов ЦП., то есть под контролем программы находятся флаги микропроцессора, определяющие знаки исходных данных и результата, наличие переноса из байта, переполнения разрядной сетки и др.

Программа готова реагировать на любой из сигналов, находящихся в ее **зоне внимания**.



Пусть «**зону внимания**» программы кроме 2-х регистров данных составляет еще и регистр флагов ЦП., то есть под контролем программы находятся флаги микропроцессора, определяющие *знаки исходных данных и результата, наличие переноса из байта, переполнения разрядной сетки* и др.

Программа готова реагировать на любой из сигналов, находящихся в ее **зоне внимания**.

**Но ....**

Если во время выполнения такой программы нажать какую-либо клавишу, то эта программа «не заметит» сигнала от этой клавиши, так как он не входит в ее **«зону внимания»**.

Если попробовать включить все заслуживающие внимания события в зону внимания программы, то это сильно усложняет программы и требует большой их избыточности.

**Такое решение практически не реализуемо!!!**

Кроме того, поскольку момент наступления события заранее неизвестен, процессор в ожидании какого-либо события может находиться длительное время, и, чтобы не пропустить его появления, ЦП не может «отвлекаться» на выполнение другой работы.

Режим сканирования ожидаемого события связан с большими потерями времени ЦП на ожидание.

**Режим может быть использован в системах с малым числом возможных событий: монитор событий (основная часть программы) + процедуры-обработчики событий. Практически НЕ ПРИМЕНИМ для многозадачных ОС общего назначения.**



Для того чтобы ЦП, выполняя свою работу, имел возможность реагировать на **события**, происходящие **вне его зоны внимания**, и наступления которых он «**не ожидает**», существует

## ***СИСТЕМА ПРЕРЫВАНИЙ ЭВМ***



Идея прерываний была предложена в середине 50-х годов.

Основные цели введения прерываний:

1. Реализация асинхронного режима работы.
2. Распараллеливание работы отдельных устройств вычислительной системы.





**Прерывания** представляют собой механизм, позволяющий координировать параллельное функционирование отдельных устройств вычислительной системы и реагировать на особые состояния, возникающие при работе процессора.

С точки зрения функционирования операционной системы **прерывание**— это **принудительная передача управления** от выполняемой программы к **программе обработки прерывания**, происходящая при возникновении определенного события.

**Структуры систем прерывания** (в зависимости от аппаратной архитектуры) могут быть самыми разными, но все они **имеют одну общую особенность** — **прерывание непременно влечет за собой изменение порядка выполнения команд процессора**.



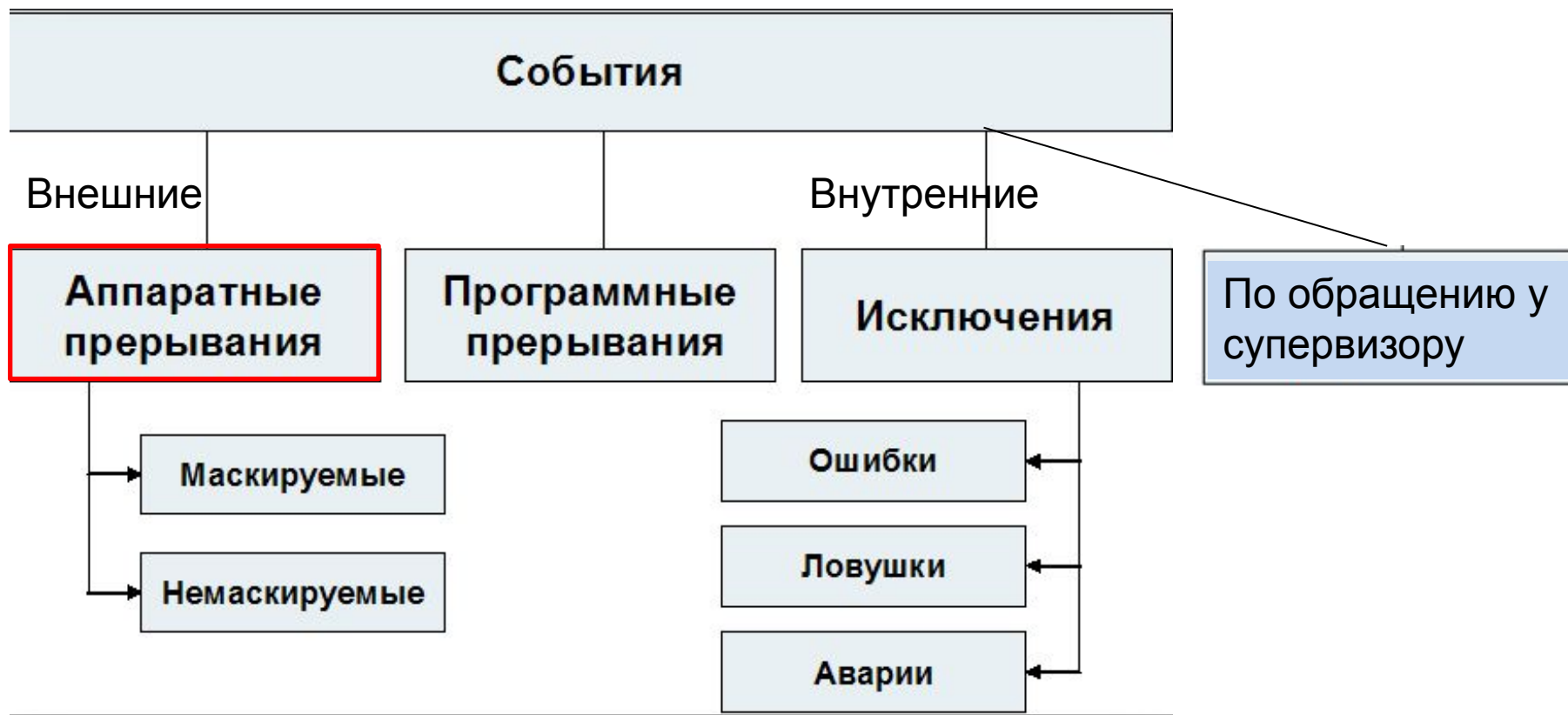
Механизм прерываний реализуется **аппаратно-программными средствами** и **независимо от архитектуры вычислительной системы** включает 7 шагов.

Шаги **1-3** реализуются **аппаратно**,  
шаги **4-7** – **программно**.



# 7 шагов механизма прерываний

- 1. Установление факта прерывания** (прием сигнала на прерывание) и идентификация прерывания (в операционных системах иногда осуществляется повторно, на шаге 4).
- 2. Запоминание состояния прерванного процесса.** Состояние процесса определяется, прежде всего, значением счетчика команд, содержимым регистров процессора и может включать также спецификацию режима (например, режим пользовательский или привилегированный) и другую информацию.
- 3. Аппаратная передача управления** подпрограмме обработки прерывания.
- 4. Сохранение информации о прерванной программе, которую не удалось спасти на шаге 2 с помощью действий аппаратуры.** В некоторых вычислительных системах предусматривается запоминание довольно большого объема информации о состоянии прерванного процесса.
- 5. Обработка прерывания.** Эта работа может быть выполнена той же подпрограммой, которой было передано управление на шаге 3, но в ОС чаще всего она реализуется путем последующего вызова соответствующей подпрограммы.
- 6. Восстановление информации, относящейся к прерванному процессу** (этап, обратный шагу 4).
- 7. Возврат в прерванную программу**



**Внешние** прерывания вызываются внешними по отношению к микропроцессору событиями, асинхронными по отношению к решаемой процессором задаче.

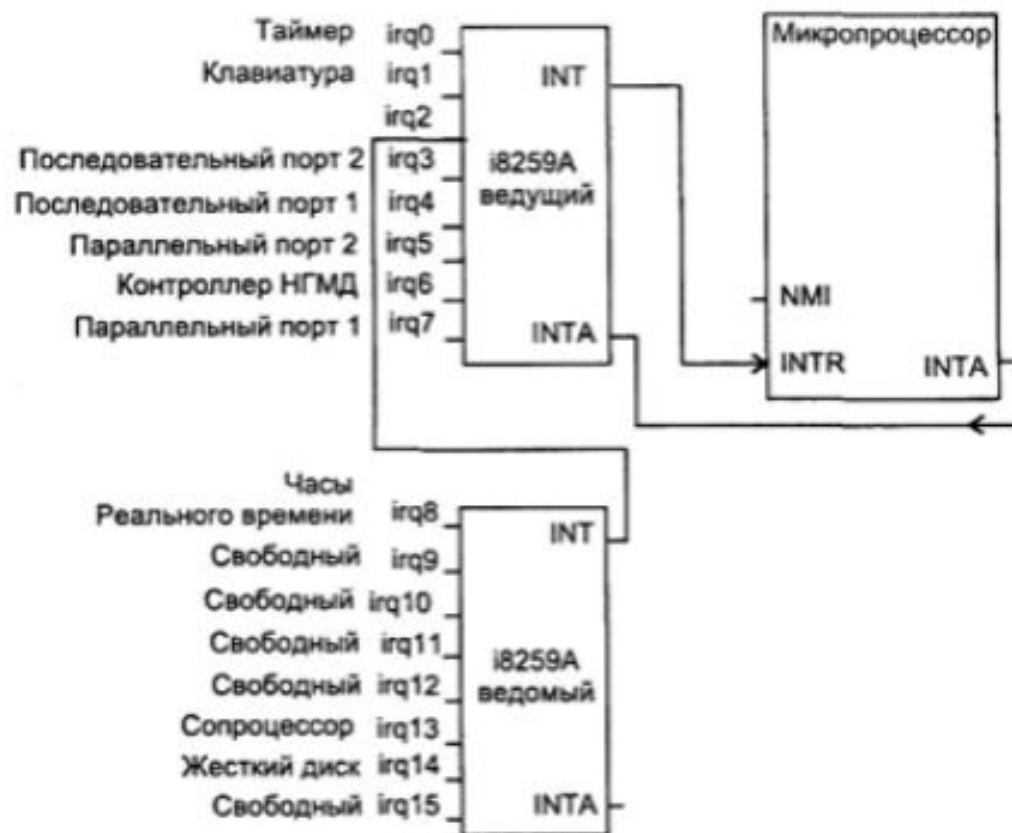
**Внутренние** прерывания и исключения вызываются событиями, которые связаны с работой процессора и являются синхронными с его операциями.



**Внешние** прерывания вызываются внешними по отношению к микропроцессору событиями, асинхронными по отношению к решаемой процессором задаче.

*Примеры:*

- прерывания от таймера;
- прерывания от внешних устройств (прерывания по вводу/выводу);
- прерывания по нарушению питания;
- прерывания с пульта оператора вычислительной системы;
- прерывания от другого процессора или другой вычислительной системы.



выводы микропроцессора :

**INTR** - вывод для входного сигнала внешнего прерывания.

**INTA** - вывод микропроцессора для выходного сигнала, подтверждения в получении сигнала прерывания микропроцессором.

**NMI** - вывод микропроцессора для входного сигнала немаскируемого прерывания.

# Таблица аппаратных прерываний, расположенных в порядке приоритета

Номер	Описание
08h	IRQ0 - прерывание интервального таймера, возникает 18,2 раза в секунду
09h	IRQ1 - прерывание от клавиатуры. Генерируется при нажатии и при отжатии клавиши. Используется для чтения данных с клавиатуры
0Ah	IRQ2 - используется для каскадирования аппаратных прерываний в машинах класса AT
70h	IRQ8 - прерывание от часов реального времени
71h	IRQ9 - прерывание от контроллера EGA
72h	IRQ10 - зарезервировано
73h	IRQ11 - зарезервировано
74h	IRQ12 - зарезервировано
75h	IRQ13 - прерывание от математического сопроцессора
76h	IRQ14 - прерывание от контроллера жесткого диска
77h	IRQ15 - зарезервировано
0Eh	IRQ3 - прерывание асинхронного порта COM2
0Ch	IRQ4 - прерывание асинхронного порта COM1
0Dh	IRQ5 - прерывание от контроллера жесткого диска для XT
0Eh	IRQ6 - прерывание генерируется контроллером флоппи-диска после завершения операции
0Fh	IRQ7 - прерывание принтера. Генерируется принтером, когда он готов к выполнению очередной операции



**Исключения** вызываются событиями, которые связаны с работой процессора и являются синхронными с его операциями.

*Примеры:*

- при нарушении адресации (в адресной части выполняемой команды указан запрещённый или несуществующий адрес, обращение к отсутствующему сегменту или странице при организации механизмов виртуальной памяти);
- при наличии в поле кода операции незадействованной двоичной комбинации;
- при делении на нуль;
- при переполнении или исчезновении порядка;
- при обнаружении ошибок чётности, ошибок в работе различных устройств аппаратуры средствами контроля.





**Программные** прерывания относятся к внутренними событиями.

Эти прерывания происходят по соответствующей команде прерывания, то есть по этой команде процессор осуществляет практически те же действия, что и при обработке исключений.

**Прерывания при обращении к супервизору ОС.**

В некоторых компьютерах часть команд может использоваться только ОС в режиме ядра.

Пользовательские программы, выполняющиеся в пользовательском режиме, привилегированные команды выполнять не могут. При попытке использовать команду, запрещённую в данном режиме, происходит внутреннее прерывание и управление передаётся супервизору ОС. К привилегированным командам относятся и *команды переключения режима работы* центрального процессора.



Сигналы, вызывающие прерывания и исключения, формируются вне процессора или в самом процессоре; они могут возникать одновременно.

Выбор одного из них для обработки осуществляется на основе приоритетов, приписанных каждому типу прерывания или в соответствии с заложенной разработчиками системы прерываний неизменной последовательностью обработки.

Очевидно, что прерывания от схем контроля процессора должны обладать наивысшим приоритетом (если аппаратура работает неправильно, то не имеет смысла продолжать обработку информации).



Обычно внешние прерывание обрабатывается только после завершения выполнения текущей команды.

Наличие сигнала прерывания не обязательно должно вызывать прерывание исполняющейся программы.

Процессор может обладать двумя средствами защиты от прерываний:

- отключение системы прерываний,
- маскирование (запрет) отдельных сигналов прерывания.

Программно-аппаратное управление этими средствами позволяет операционной системе регулировать обработку сигналов прерывания, заставляя процессор *обрабатывать их сразу по приходу, откладывать их обработку на некоторое время или полностью игнорировать.*



Программное управление специальными регистрами маски (маскирование сигналов прерывания) позволяет реализовать различные дисциплины обслуживания:

- с относительными приоритетами;
- с абсолютными приоритетами.



Обслуживание принятого на обработку запроса не прерывается даже при наличии запросов с более высокими приоритетами.

После окончания обслуживания данного запроса обслуживается запрос с наивысшим приоритетом.

Для организации такой дисциплины необходимо в программе обслуживания данного запроса наложить маски на все остальные сигналы прерывания или просто отключить систему прерываний.



*Всегда обслуживается прерывание с наивысшим приоритетом!* Обслуживание принятого на обработку запроса

прерывается только при наличии запросов с более высокими приоритетами.

После окончания обслуживания самого приоритетного запроса из поступивших к этому моменту запросов на обслуживание выбирается запрос с наивысшим приоритетом.

Для реализации этого режима необходимо на время обработки прерывания замаскировать все запросы с более низким приоритетом.

При этом возможно *многоуровневое прерывание*, то есть прерывание программ обработки прерываний.



- Внешние прерывания делятся на немаскируемые и маскируемые.
- Запрос немаскируемого прерывания, поступающий на вход NMI процессора, должен быть обслужен сразу. Он обычно сообщает о чрезвычайной ситуации, например, об обнаружении ошибки в памяти или временном понижении напряжения питания.
- Маскируемые прерывания поступают на вход INT процессора от устройств ввода-вывода, требующих обслуживания.  
Эти прерывания можно запрещать или разрешать.

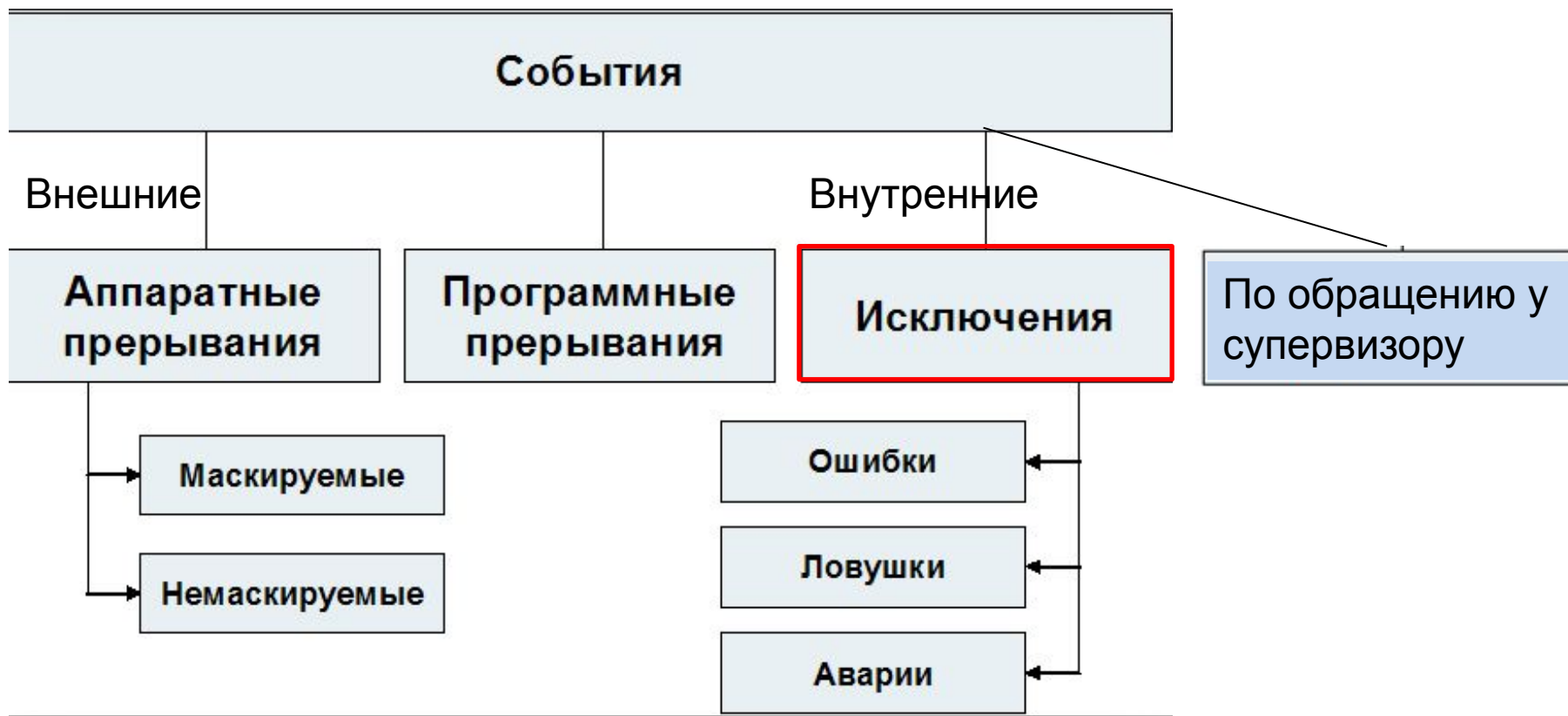


- сброс или установка соответствующих биты INTE (Interrupt Enable) регистров состояний интерфейсных блоков;
- -установка или сброс соответствующих биты регистра маски контроллера прерываний
- выполнение команд CLI (Clear Interrupt) или STI (Set Interrupt), соответственно сбрасывающей или устанавливающей флаг IF регистра флагов процессора.

## ВАЖНО!!!

Прерывания от стандартных устройств ввода-вывода **запрещать** рекомендуется на **очень короткие промежутки времени**, необходимые для выполнения критических участков программы.





**Внешние** прерывания вызываются внешними по отношению к микропроцессору событиями, асинхронными по отношению к решаемой процессором задаче.

**Внутренние** прерывания и исключения вызываются событиями, которые связаны с работой процессора и являются синхронными с его операциями.



- **ОТКАЗЫ** (faults) - выявляются и обслуживаются перед выполнением команды. Они могут иметь место в виртуальной системе памяти, когда процессор обращается к отсутствующим в физической оперативной памяти странице или сегменту. В процессе обработки такого исключения операционная система обращается к странице или сегменту на диске, а микропроцессор перезапускает команду;
- **ЛОВУШКИ** (traps) – прерывания, возникающие непосредственно после выполнения команды автоматически или после выполнения соответствующей команде INT;
- **АВАРИЯ** или **ВЫХОДЫ ИЗ ПРОЦЕССА** (aborts) - возникают при обнаружении крупных, неисправимых ошибок в системных таблицах или оборудовании.



- 1h - ОТЛАДКА возникает при пошаговом выполнении; при обращении к сегменту состояния задачи (TSS) с установленным битом ловушки (бит T); при достижении контрольной точки, установленной в регистрах DR0-DR3.
- 3h - КОНТРОЛЬНАЯ ТОЧКА по существу, является программным прерыванием, но в отличие от остальных вызывается однобайтовой командой INT3.



- 0h - ОШИБКА ДЕЛЕНИЯ генерируется сразу после выполнения команд деления DIV и IDIV, если формат частного превышает формат получателя или в случае деления на 0.
- 4h - ПЕРЕПОЛНЕНИЕ генерируется по однобайтовой команде INTO, если установлен флаг переполнения OF при выполнении сложения или вычитания.
- 5h - ПРЕВЫШЕНИЕ ГРАНИЦЫ МАССИВА происходит при выполнении команды BOUND, если содержимое адресуемого регистра выходит за указанные пределы.
- 6h - НЕРАЗРЕШЕННЫЙ КОД КОМАНДЫ попытка выполнения команды с несуществующим кодом или неразрешенным операндом; неверное использование префикса LOCK.
- 7h - ПРОЦЕССОР ВЫЧИСЛЕНИЙ С ПЛАВАЮЩЕЙ ТОЧКОЙ НЕДОСТУПЕН возникает если обращение к сопроцессору



- Ah - НЕРАЗРЕШЕННЫЙ СЕГМЕНТ TSS возникает при переключении задач при неверном TSS; отсутствии соответствующей LDR; нарушении защиты памяти; неверных значениях сегментных регистров.
- Bh - ОТСУТСТВИЕ СЕГМЕНТА обращение к сегменту, находящемуся вне ОП.
- Ch - ОШИБКА ОБРАЩЕНИЯ К СТЕКУ возникает при обращении к сегменту стека, отсутствующему в ОП; нарушение границы стека.
- Dh - НАРУШЕНИЕ ОБЩЕЙ ЗАЩИТЫ нарушение защиты при обращении к памяти или порту; нарушение границ сегментов при обращении к таблице дескрипторов; неправильная загрузка регистра управления процессором; запись в ячейку доступную только для чтения; переключение на занятую задачу; передача управления сегменту данных; превышения максимальной длины команды (15 байт).



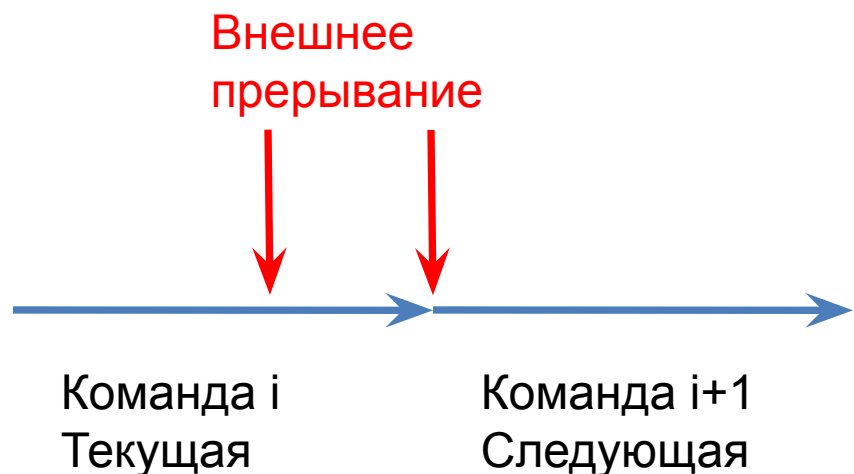
- Eh - ОТСУТСТВИЕ ДОСТУПА К СТРАНИЦЕ имеет место при страничной адресации если текущая процедура не имеет достаточного уровня привилегий для вызова указанной страницы или происходит обращение к несуществующему разделу или странице.
- 10h - ОШИБКА ОПЕРАЦИИ С ПЛАВАЮЩЕЙ ТОЧКОЙ возникает при обнаружении ошибки, если реакция на ошибки разрешена.
- 11h - ОШИБКА ВЫРАВНИВАНИЯ возможна, если возникла ошибка и реакция разрешена. Требование по выравниванию: адрес должен быть кратен длине выбираемой информации.



- 8h - ДВОЙНАЯ ОШИБКА фиксируется, если при обслуживании исключений Ah..Dh возникло новое исключение с номером отличным от Eh. Если данное исключение возникает дважды, то работа процессора блокируется до сигнала "сброс" или до возникновения немаскируемого прерывания.



# Последовательность проверки исключений и прерываний

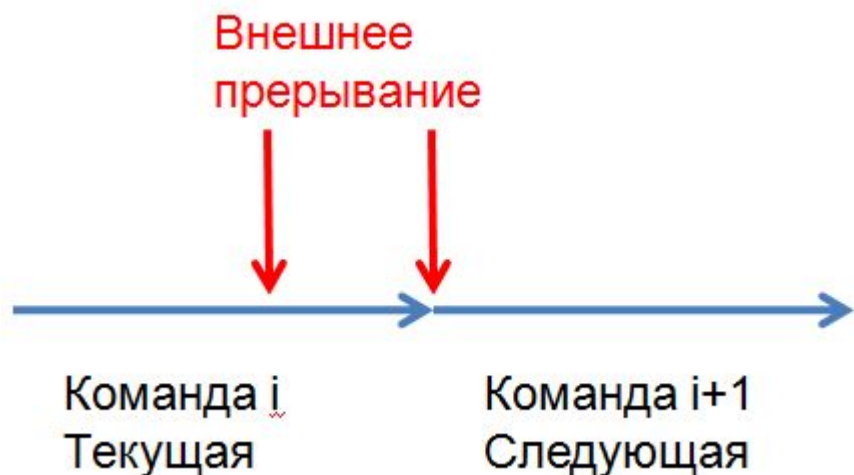


Запросы на обработку исключений генерируются внутренними блоками микропроцессора.

Одна команда может быть причиной нескольких исключений, однако в каждый момент будет возникать только одно.

Процессор будет обрабатывать возникающие ситуации и перезапускать команду до полного ее выполнения.





1. Проверяются ловушки только что завершенной команды (пошаговый режим, прерывания по значению результата);

2. Проверяются отказы следующей команды по значениям регистров отладки;

3. Проверяется наличие запросов **внешних прерываний**;

- Внешние прерывания проверяются и обрабатываются **между выполнением** команд программы.
- **Немаскируемое** прерывание имеет **более высокий приоритет**, чем **маскируемое**.



4. Проверяются отказы сегментации, мешающие выборке следующей команды;

5. Проверяются отказы страничной организации (исключение Eh);

6. Проверяются отказы дешифрации (длина команды, возможность выборки операндов);

7. Проверяется возможность выполнения операции с плавающей точкой;

8. Проверяется наличие ошибки при операции с плавающей точкой, если проверка включена.

Основные режимы работы  
микропроцессоров  
с архитектурой IA-32 и x86-64

Реальный режим

Разрядность – 16  
Вид памяти - физическая  
Объем памяти – 1 Мб  
Адресация памяти –  
прямая адресация  
сегментов памяти  
(регистры адресов  
сегментов)

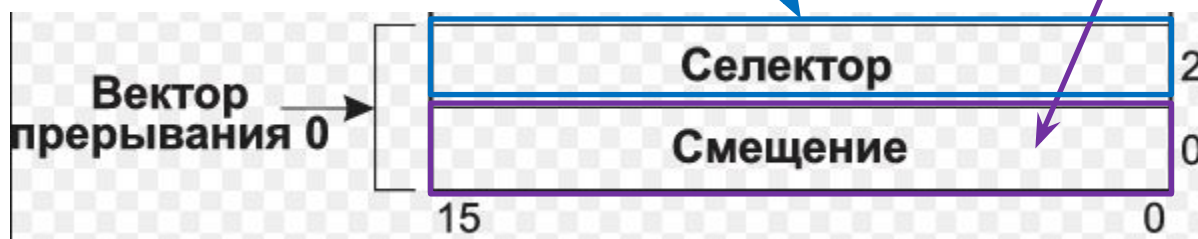
Защищенный режим

Разрядность – 32/64  
Вид памяти – виртуальная  
Объем памяти – 4 Гб/  $2^{64}$   
байт Адресация памяти –  
косвенная адресация  
сегментов памяти  
(селекторные регистры,  
адресующие дескрипторы  
сегментов)

В реальном режиме работы система прерываний использует понятие «**вектор прерывания**».

Термин «вектор прерываний» используется потому, что для указания адреса используется не одно значение, а два, то есть мы имеем дело не со скалярной величиной, а с «векторной».

Каждый вектор прерываний состоит из 4 байтов: первые два байта содержат новое значение для **регистра IP**, следующие два – новое значение **регистра CS**.





- **Таблица векторов прерываний** содержит **256** векторов прерываний и занимает 1024 байта.
- В процессоре i8086 эта таблица располагается на адресах 00000-003FFH.

Расположение этой таблицы в процессорах i80286 и старше определяется значением регистра IDTR – Interrupt Descriptor Table Register.

При включении или сбросе процессора i80x86 этот регистр обнуляется. Однако при необходимости *можно в регистре IDTR указать смещение и, таким образом, перейти на новую таблицу векторов прерываний.*

- Таблица векторов прерываний заполняется (инициализируется) при запуске системы, но в принципе **может быть изменена или перемещена.**

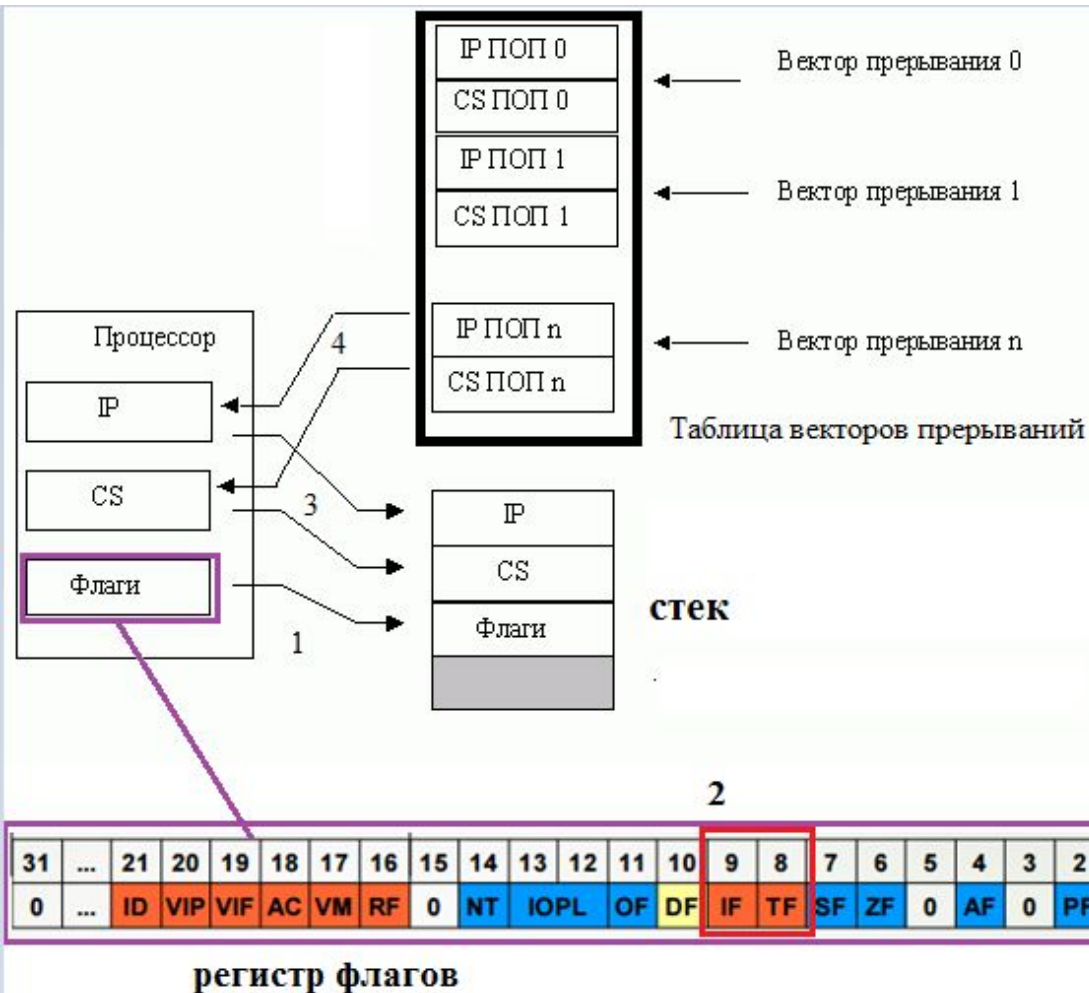


# Как найти нужный вектор прерываний в таблице

Каждый вектор прерывания имеет свой номер, называемый *номером прерывания*, который указывает его место в таблице.

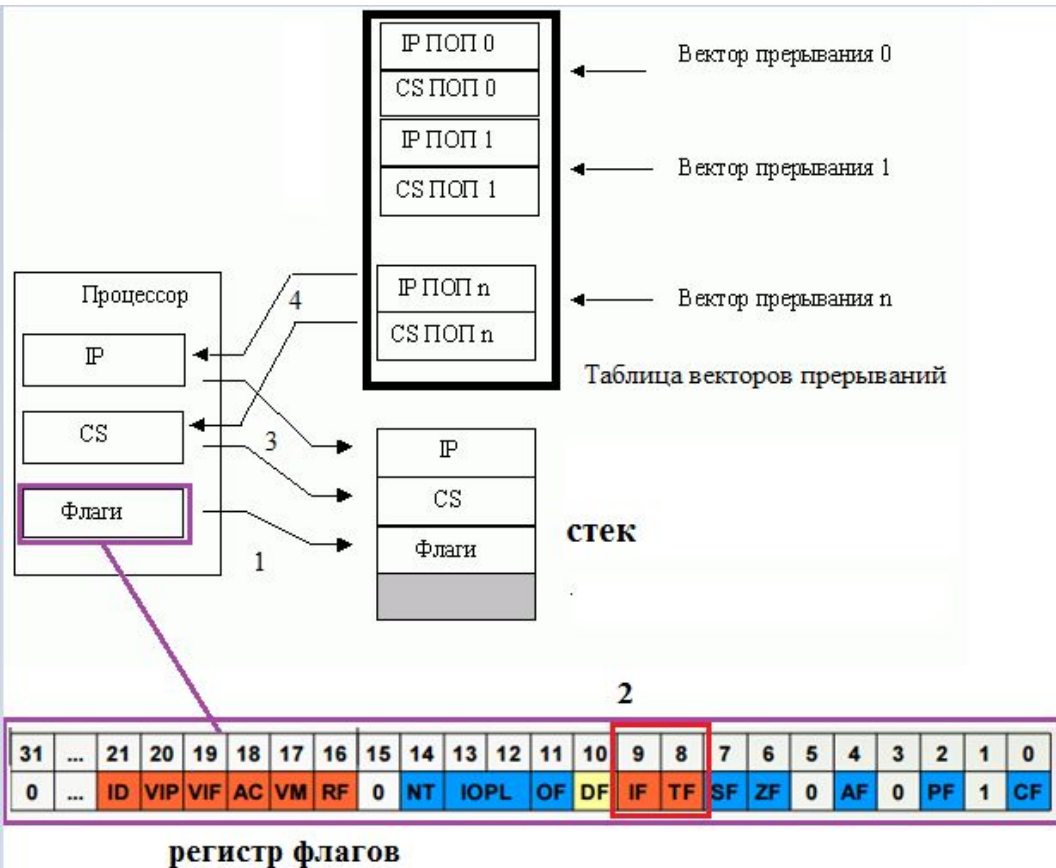
Этот номер, помноженный на четыре (сдвиг на два разряда влево и заполнение освободившихся битов нулями), и сложенный с содержимым регистра IDTR, дает абсолютный адрес первого байта вектора в оперативной памяти.

# Процедура перехода на подпрограмму обработки прерывания или исключения



1. В стек помещается **регистр флагов**.
2. Флаг включения/выключения прерываний IF и флаг трассировки TF, находящиеся в регистре флагов, обнуляются для блокировки других маскируемых прерываний и исключения пошагового режима исполнения команд.
3. Значения **регистров CS и IP** сохраняются в стеке вслед за регистром флагов.
4. Вычисляется адрес вектора прерывания, и из вектора, соответствующего номеру прерывания, загружаются новые значения регистров **IP** и **CS**. Когда системная подпрограмма принимает управление, она может снова разрешить маскируемые прерывания командой **STI** (set interrupt flag, установить флаг прерываний), которая переводит флаг IF в состояние 1, что разрешает микропроцессору вновь реагировать на прерывания, инициируемые внешними устройствами, поскольку стековая организация позволяет вложение прерываний друг в друга.

Закончив работу, подпрограмма обработки прерывания должна выполнить команду **IRET** (interrupt return), которая извлекает из стека три 16-битовых значения и загружает их в указатель команд IP, регистр сегмента команд CS и регистр флагов соответственно. Таким образом, процессор сможет продолжить работу программы с того места, где она была прервана.



1. В стек помещается **регистр флагов**.
2. Флаг включения/выключения прерываний IF и флаг трассировки TF, находящиеся в регистре флагов, обнуляются для блокировки других маскируемых прерываний и исключения пошагового режима исполнения команд.
3. Значения **регистров CS и IP** сохраняются в стеке вслед за регистром флагов.
4. Вычисляется адрес вектора прерывания, и из вектора, соответствующего номеру прерывания, загружаются новые значения регистров **IP** и **CS**. Когда системная подпрограмма принимает управление, она может снова разрешить маскируемые прерывания командой **STI** (set interrupt flag, установить флаг прерываний), которая переводит флаг IF в состояние 1, что разрешает микропроцессору вновь реагировать на прерывания, инициируемые внешними устройствами, поскольку стековая организация позволяет вложение прерываний друг в друга.

**0.1.** Контроллер прерываний получает заявку от определенного периферийного устройства и, соблюдая схему приоритетов, генерирует сигнал INTR (interrupt request), который является входным для микропроцессора.

**0.2.** Микропроцессор проверяет флаг IF в регистре флагов. Если он установлен в 1, то прерывания разрешены и система переходит к шагу 0.3. В противном случае прерывания отключены и работа процессора не прерывается.

**0.3.** Микропроцессор генерирует сигнал INTA (подтверждение прерывания). В ответ на этот сигнал контроллер прерывания посылает по шине данных номер прерывания. После этого выполняется описанная ранее процедура передачи управления соответствующей программе обработки прерывания.



Работа механизма реакции на *прерывания* и *особые ситуации* в защищенном режиме базируется на специальной **таблице дескрипторов прерываний** (*Interrupt Descriptor Table – IDT*).

**Базовый адрес (BASE)** этой таблицы и ее **размер (LIMIT)** заносятся в регистр IDTR с помощью команды LIDT (*Load IDTR*)

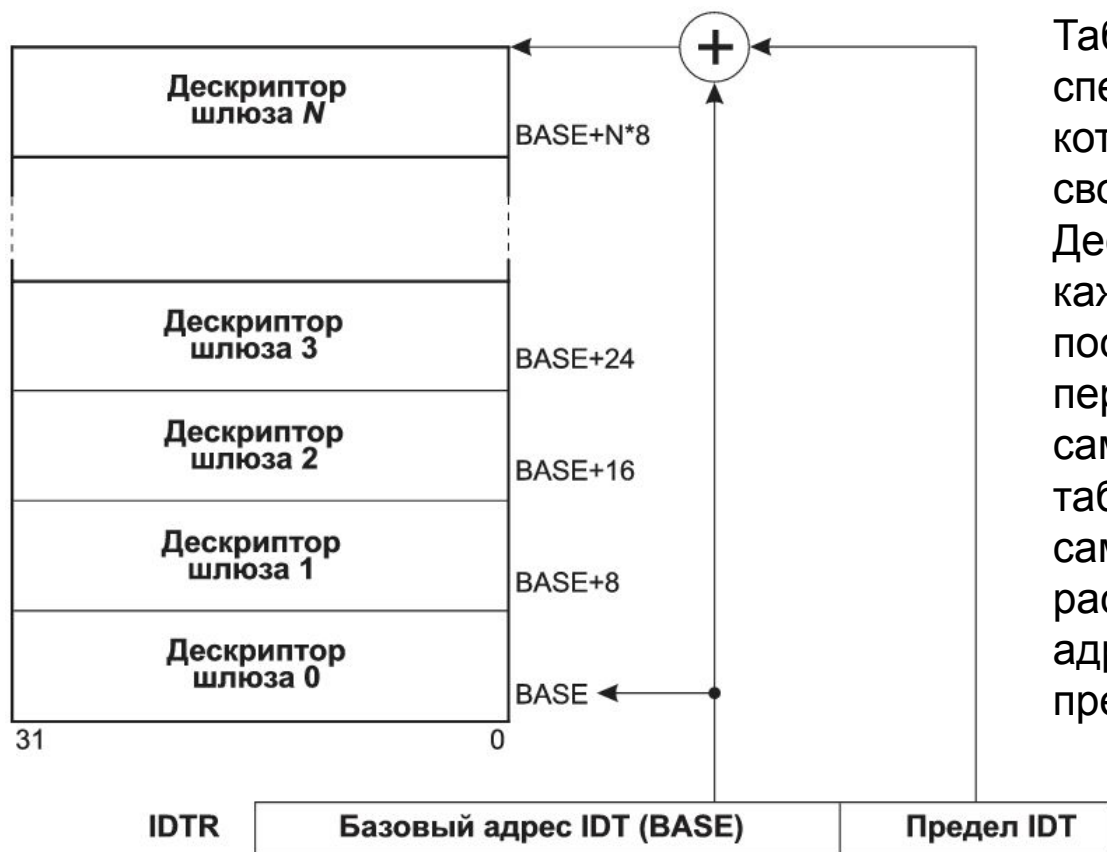


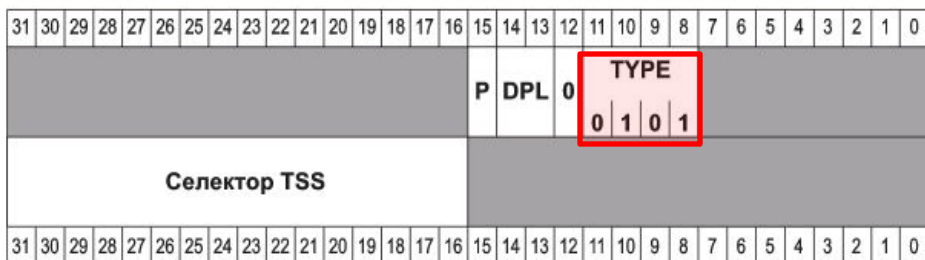
Таблица **IDT** содержит до 256 специальных дескрипторов, каждый из которых соответствует своему *прерыванию*. Дескрипторы занимают по 8 байт каждый и располагаются в таблице последовательно. Самый первый *дескриптор*, расположенный по самому младшему адресу в начале таблицы соответствует прерыванию 0, а самый последний *дескриптор*, расположенный по самому старшему адресу в таблице, соответствует прерыванию 255.



- В качестве дескрипторов таблицы IDT могут использоваться дескрипторы разного типа – **шлюза прерывания**, **шлюза ловушки** и **шлюза задачи**. Поведение процессора при поступлении запроса на прерывание зависит, как от типа применяемого для данного прерывания дескрипторов, так и от текущего уровня привилегий задачи.
- В случае, если в качестве дескриптора прерывания используется **шлюз задачи**, процессор при поступлении запроса на прерывание выполняет все действия по переключению на новую задачу, сохраняя текущее состояние в **TSS** старой задачи и загружая новые значения в регистры из **TSS** задачи обработчика прерывания. Применение такого шлюза оправдано для обработки ошибок и сбоев, которые не могут быть обработаны в рамках текущей задачи (например, ошибки, возникающие при переключении задач).
- Если в качестве дескриптора прерывания используется **шлюз ловушки** или **шлюз прерывания**, то переключения задач не происходит и программа обработчик функционирует в рамках той же задачи, что и прерванный код.

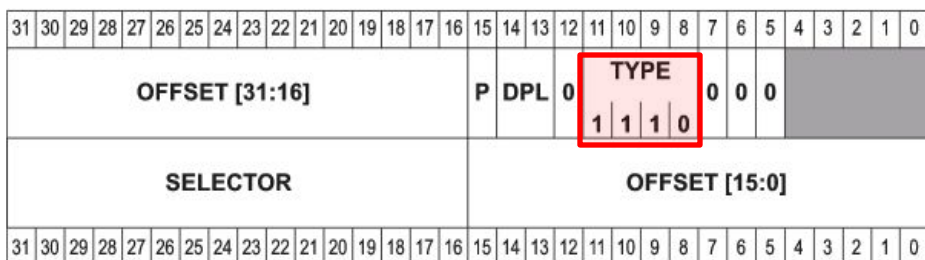
**Отличие шлюза прерывания** состоит лишь в том, что процессор автоматически сбрасывает флаг **EFLAGS.IF** при передаче управления в обработчик, чем обеспечивается маскирование внешних прерываний. В последующем, при возврате из процедуры обработчика прежнее значение регистра EFLAGS восстанавливается из стека.

<http://www.club155.ru/x86exceptions-protectedmode>



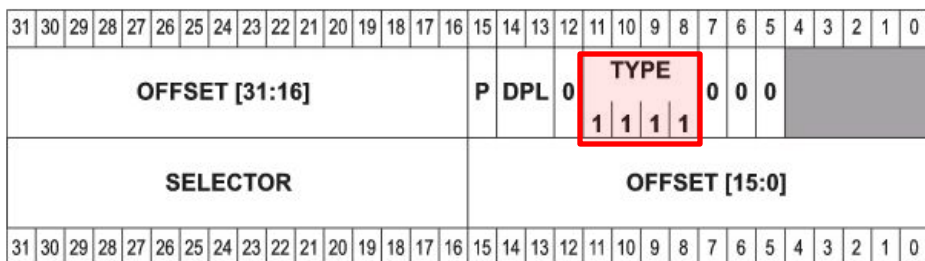
■ Зарезервировано.

Формат дескриптора шлюза задачи



■ Зарезервировано.

Формат дескриптора 32-битного шлюза прерывания



■ Зарезервировано.

Формат дескриптора 32-битного шлюза ловушки

**P** (Присутствие)

4 Бит присутствия сегмента в физической памяти.

0 **DPL** (*Уровень привилегий дескриптора*)

Задаёт уровень привилегий сегмента или шлюза, который (уровень) используется механизмом защиты для контроля доступа к сегменту или шлюзу.

**TYPE** (Тип, биты 11..8 старшего двойного слова дескриптора)

4 Поле тип содержит **биты**, которые **задают**

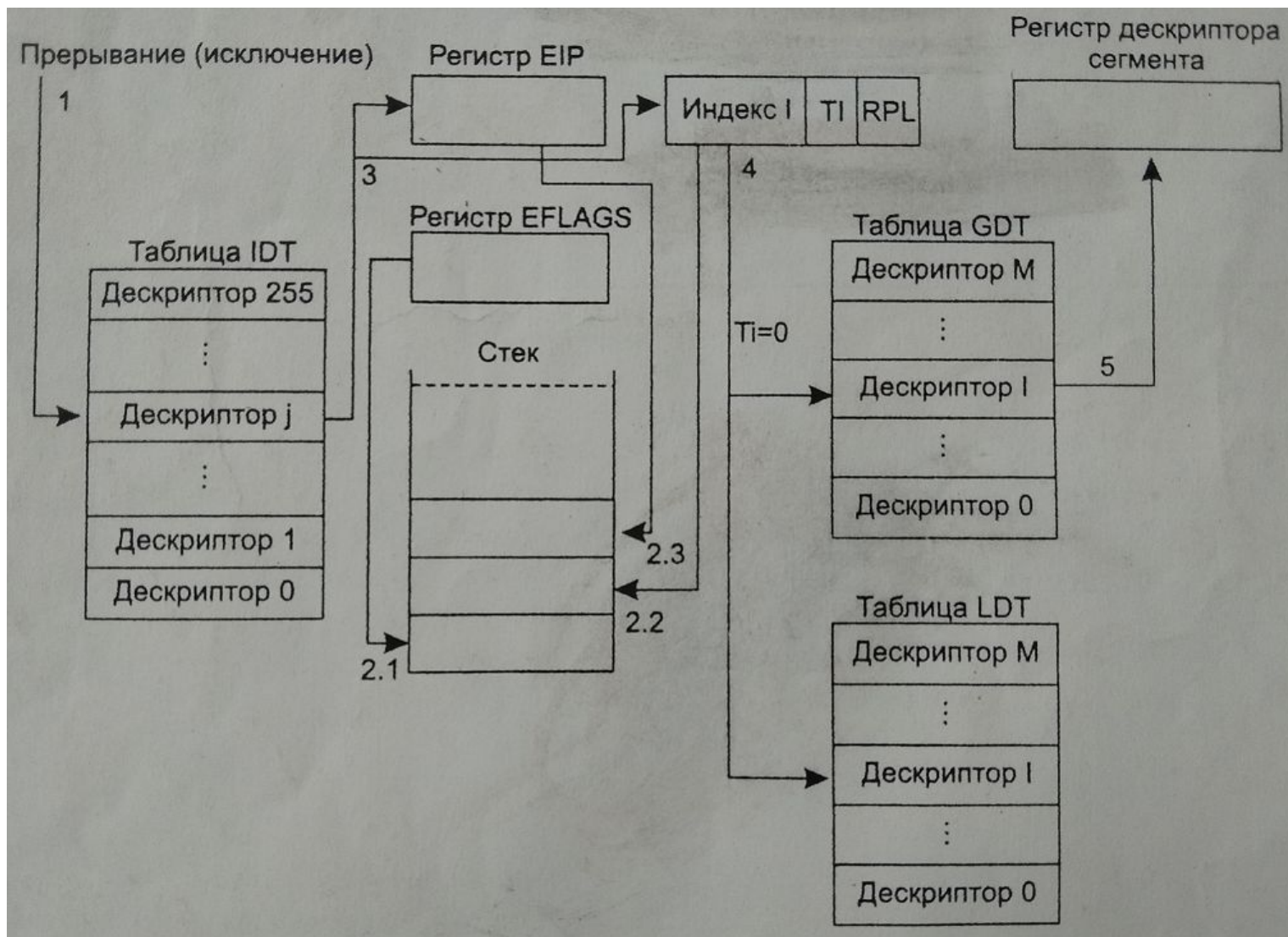
0 **назначение системного дескриптора.**

**OFFSET** (Смещение)

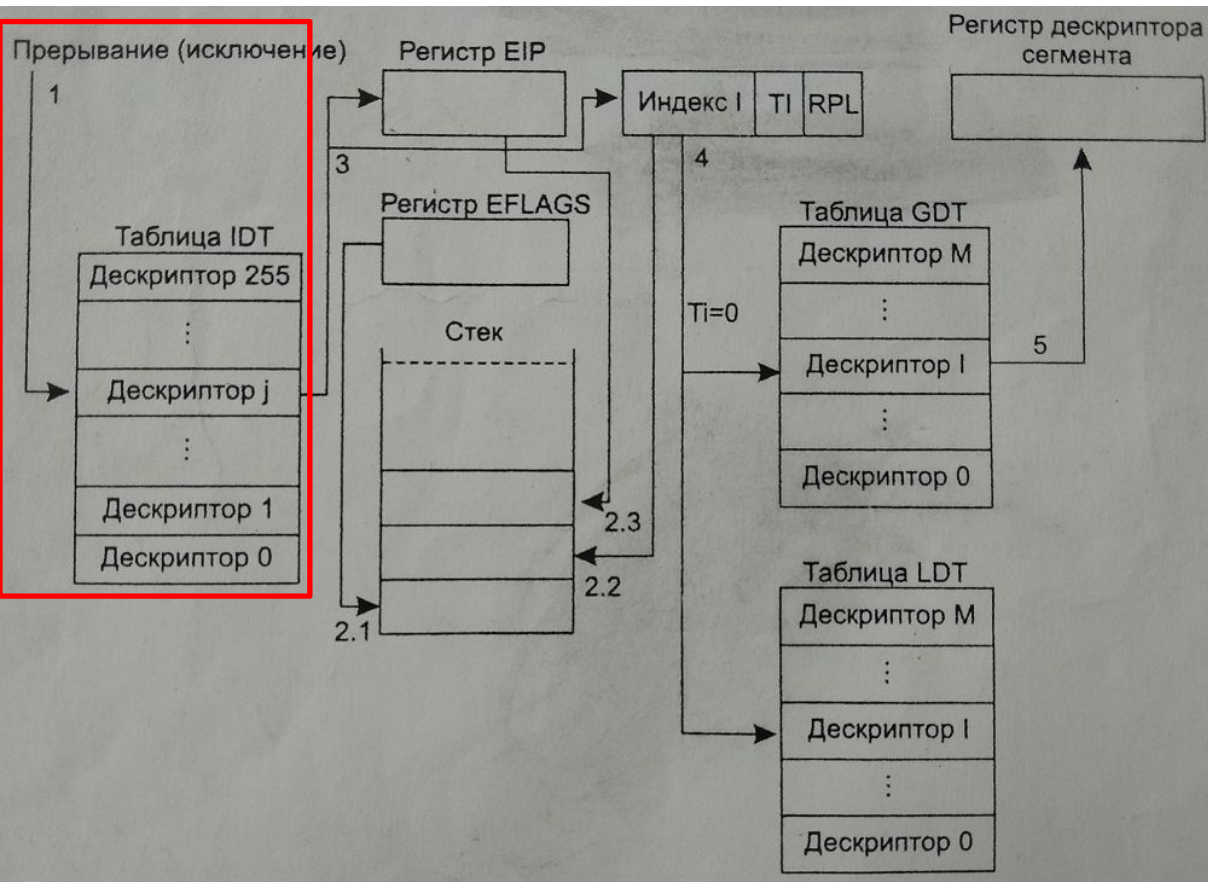
Смещение точки входа в целевом сегменте.

**SELECTOR** (Селектор)

Селектор целевого сегмента или (для шлюза задачи) селектор блока состояния задачи (TSS) целевой задачи.

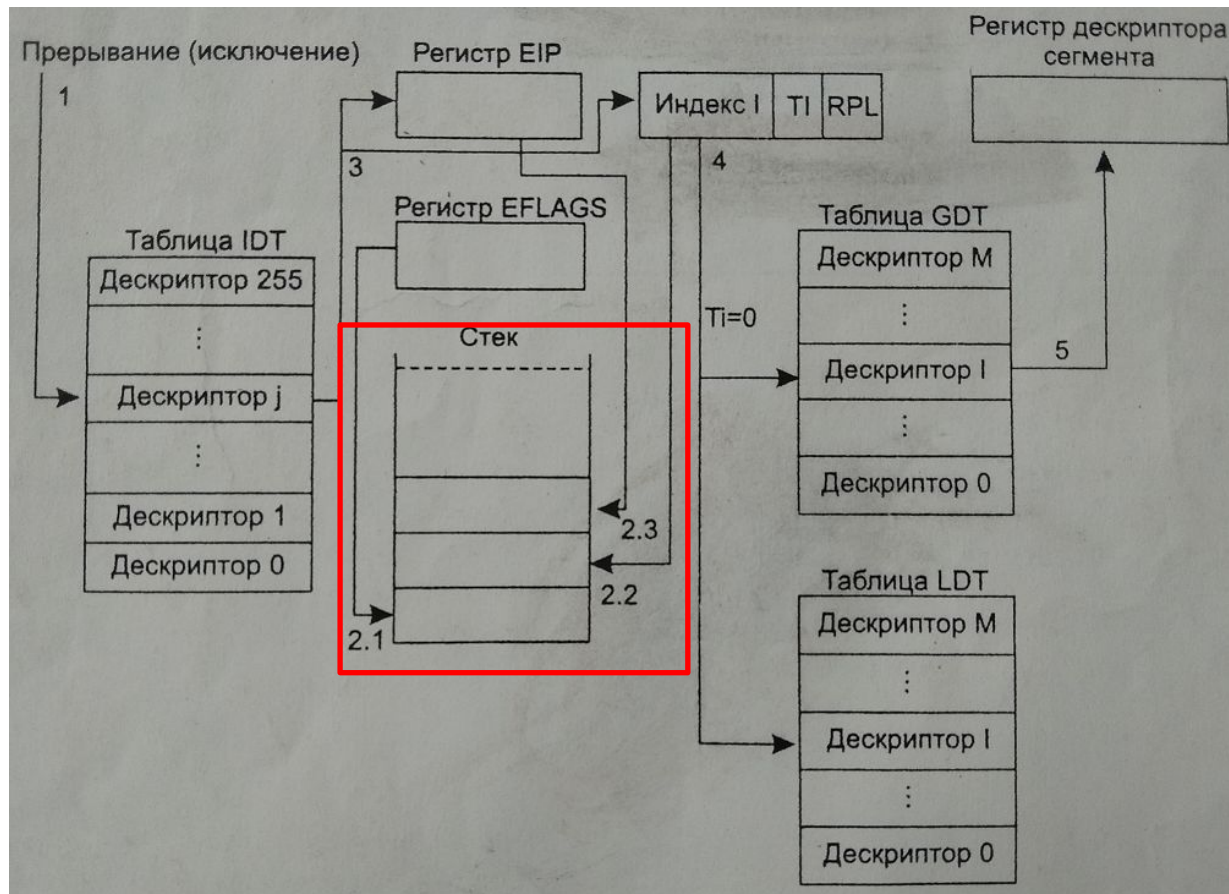


Полученный дескриптор анализируется, и если его тип соответствует *шлюзу ловушки* или *шлюзу прерывания*, то выполняются следующие действия.



1. При возникновении прерывания процессор по номеру прерывания индексирует таблицу IDT, то есть адрес соответствующего коммутатора определяется путем сложения содержимого поля адреса в регистре IDTR и номера прерывания, умноженного на 8 (справа к номеру прерывания добавляются три нуля).

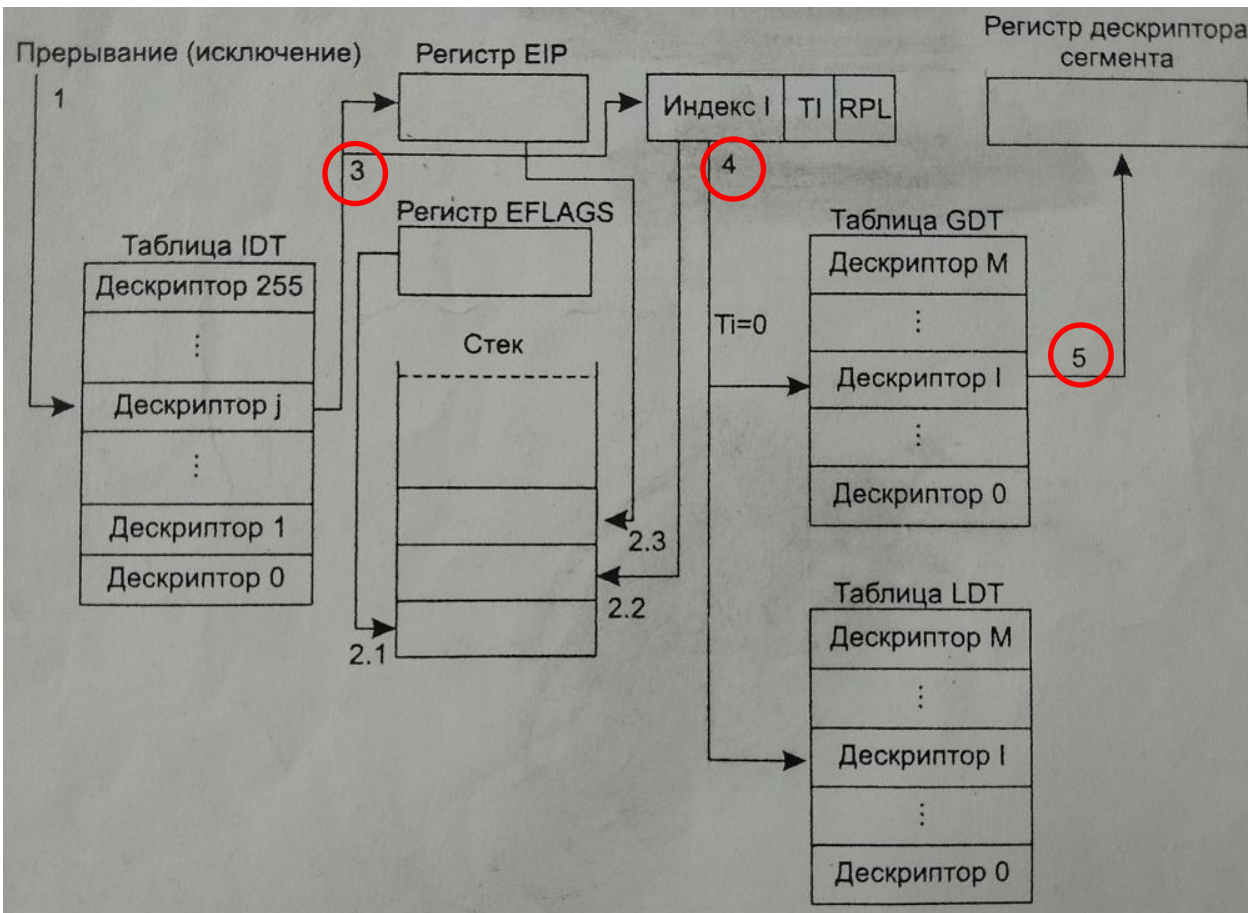
Полученный дескриптор анализируется, и если его тип соответствует *шлюзу ловушки* или *шлюзу прерывания*, то выполняются следующие действия.



2. В стек на уровне привилегий текущего сегмента кода помещаются:

- значения регистров SS и SP, если уровень привилегий DPL шлюза выше уровня привилегий ранее исполнявшегося кода (2.1);
- регистр флагов EFLAGS (2.2);
- регистры CS и IP (2.3) .

Если рассматриваемому прерыванию соответствовал *шлюз прерывания*, то запрещаются прерывания (флаг IF=0 в регистре EFLAGS). В случае *шлюз ловушки* флаг прерываний не сбрасывается и обработка новых прерываний на период обработки текущего прерывания тем самым не запрещается.

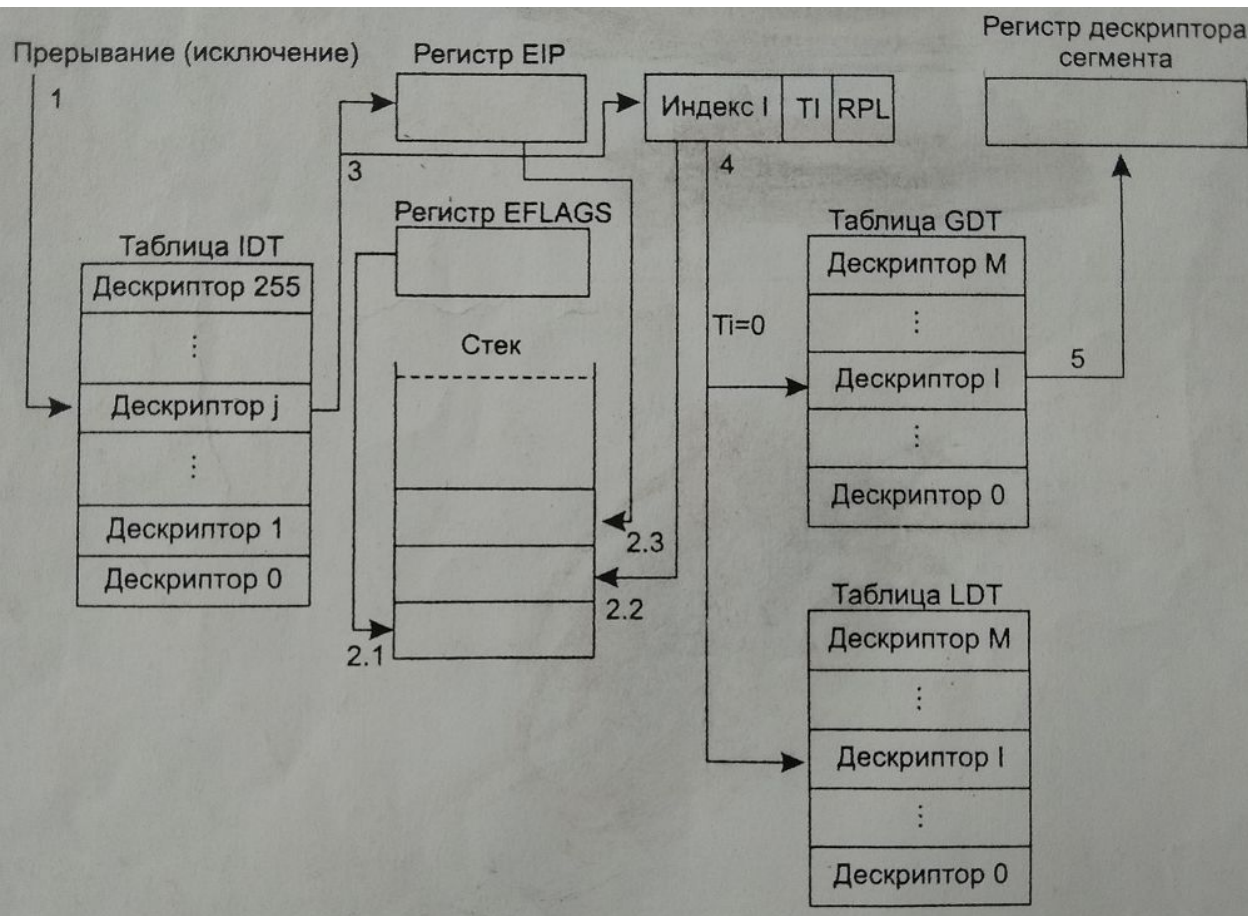


3. Поле селектора из дескриптора прерываний переписывается в селекторный *регистр CS*.

4. Поле Index этого селектора используется для индексирования таблицы дескрипторов задачи при выборе дескриптора целевого сегмента кода.

5. Дескриптор сегмента заносится в теневой регистр, а смещение относительно начала нового сегмента кода определяется **полем смещения из дескриптора прерывания**.

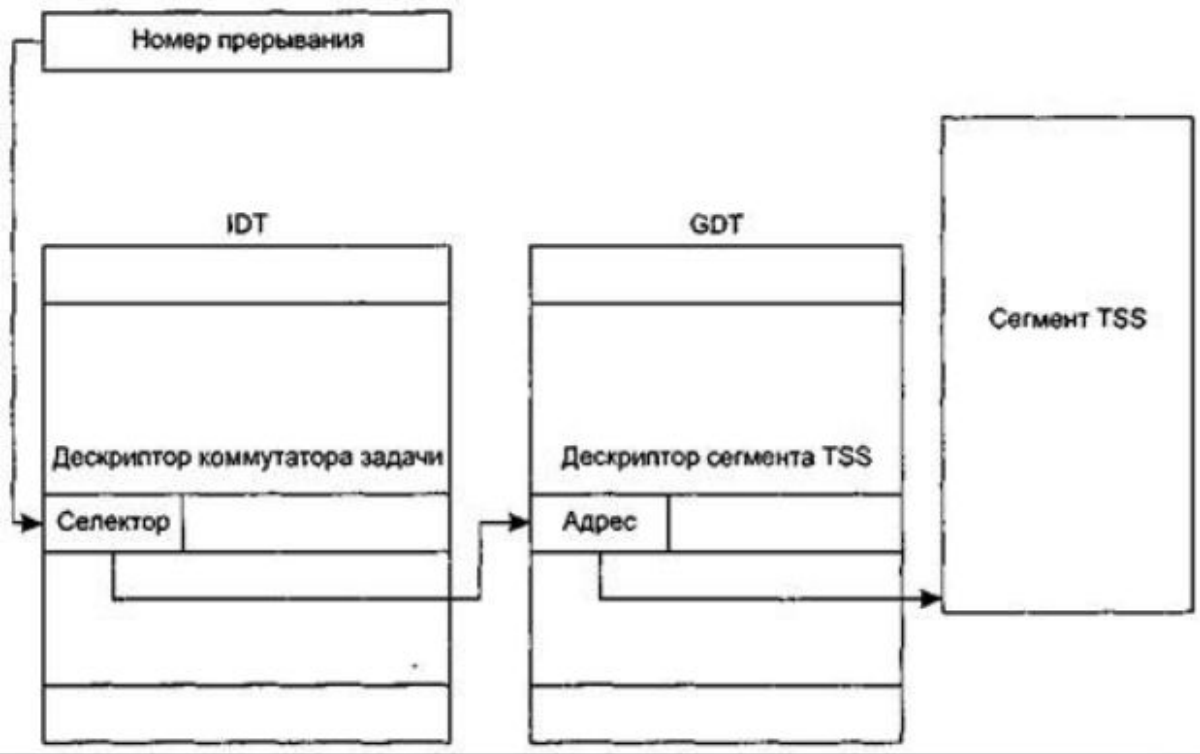
# Схема передачи управления при прерывании в контексте текущей задачи



Таким образом, в случае обработки прерываний, когда дескриптором прерываний является *шлюз ловушки* или *шлюз прерывания*, процесс остается в том же виртуальном адресном пространстве, и полной смены контекста текущей задачи не происходит. Просто происходит передача управления другому, как правило, более привилегированному коду.

Этот код создается системными программистами, а прикладные программисты его просто используют. В то же время механизмы защиты микропроцессора позволяют обеспечить недоступность этого кода для его исправления (со стороны приложений, его вызывающих) и недоступность самой таблицы дескрипторов прерываний.





Формат шлюза задачи (*task gate*) отличается от шлюзов прерывания и ловушки, прежде всего, тем, что в нем вместо селектора сегмента кода, на который передаётся управление, указывается селектор сегмента состояния задачи (TSS).

В результате осуществляется процедура перехода на новую задачу с полной сменой контекста, ибо сегмент состояния задачи полностью определяет новое виртуальное пространство и адрес начала программы, а текущее состояние прерываемой задачи аппаратно (по микропрограмме микропроцессора) сохраняется в её собственном TSS.

При этом происходит полное переключение на новую задачу.