В защищённом режиме все прерывания разделяются на два типа - обычные прерывания и исключения (exception - исключение, особый случай).
Обычное прерывание инициируется командой INT (программное прерывание) или внешним событием (аппаратное прерывание). Перед передачей управления процедуре обработки обычного прерывания флаг разрешения прерываний IF сбрасывается и прерывания запрещаются.
Исключение происходит в результате ошибки, возникающей при выполнении какой-либо команды, например, если команда пытается выполнить запись данных за пределами сегмента данных или использует для адресации селектор, который не определён в таблице дескрипторов.
По своим функциям исключения соответствуют зарезервированным для процессора внутренним прерываниям реального режима. Когда процедура обработки исключения получает управление, флаг IF не изменяется. Поэтому в мультизадачной среде особые случаи, возникающие в отдельных задачах, не оказывают влияния на выполнение остальных задач.
В защищённом режиме прерывания могут приводить к переключению задач.
Теперь перейдём к рассмотрению механизма обработки прерываний и исключений в защищённом режиме.

Таблица прерываний защищённого режима

Таблица прерываний защищённого режима является таблицей дескрипторов, которая содержит так называемые вентили прерываний, вентили исключений и вентили задач.

Таблица прерываний защищённого режима называется дескрипторной таблицей прерываний IDT (Interrupt Descriptor Table). В защищённом режиме перед передачей управления процессор производит множество проверок возможности доступа к обработчику прерывания - обеспечивает защиту. Таблица дескрипторов прерываний (IDT) в любой системе - одна. Программ (задач, процедур, приложений и пр.) - много. IDT реализуется на нулевом уровне привилегий и, следовательно, непосредственно к ней обратиться могут только программы, работающие на том же уровне. Для того, чтобы программы с других уровней (1, 2 и 3) могли пользоваться прерываниями, предусмотрены специальные системные объекты - так называемые шлюзы (gates). При вызове прерывания, процессор, прежде, чем передать управление обработчику, "опускается" через шлюз на его уровень привилегий, а после завершения обработки - "поднимается" обратно.

IDT может содержать три типа дескрипторов шлюзов:

Шлюз задачи
Шлюз прерывания
Шлюз ловушки

Шлюзы содержат указатели на обработчики прерываний и права доступа к ним. При переходе через шлюз задачи, процессор производит автоматическое переключение задач, а при переходе через шлюз прерывания или ловушки передаёт управление процедуре в контексте текущей программы. Единственное отличие прерывания от ловушки в том, что при переходе через шлюз прерывания процессор автоматически сбрасывает флаг IF в EFLAGS и тем самым не допускает генерации других прерываний и исключений на время работы обработчика, а для шлюза ловушки - не меняет состояние флага IF. Ловушки используются для отладки программ и поэтому обработка ловушки должна быть прозрачна для внешних прерываний.

Исключения и прерывания работают в основном через два типа шлюзов - задач и прерываний. Шлюз прерывания запускает обработчик в контексте текущей программы, т.е. просто передаёт управление по адресу, указанному в дескрипторе. Такой подход хорош только в простых операционных системах, когда работают заранее определённые программы, от которых не нужно защищать ядро ОС.

Шлюз задачи является более удобным и универсальным, т.к. позволяет изолировать обработчик от других программ и его рекомендуется применять в системах, где программы потенциально могут нарушить целостность ОС. Шлюз задачи заставляет процессор автоматически переключаться на новую задачу при генерации исключения. Т.к. мультизадачность мы ещё не рассматривали, то обработчики исключений реализуем пока через шлюзы прерываний и ловушек.

Шлюз задачи.

```
dw 0
dw TSS_sel ; Селектор TSS
db 0
db access_rights ; Права доступа сегмента TSS
dw 0
```

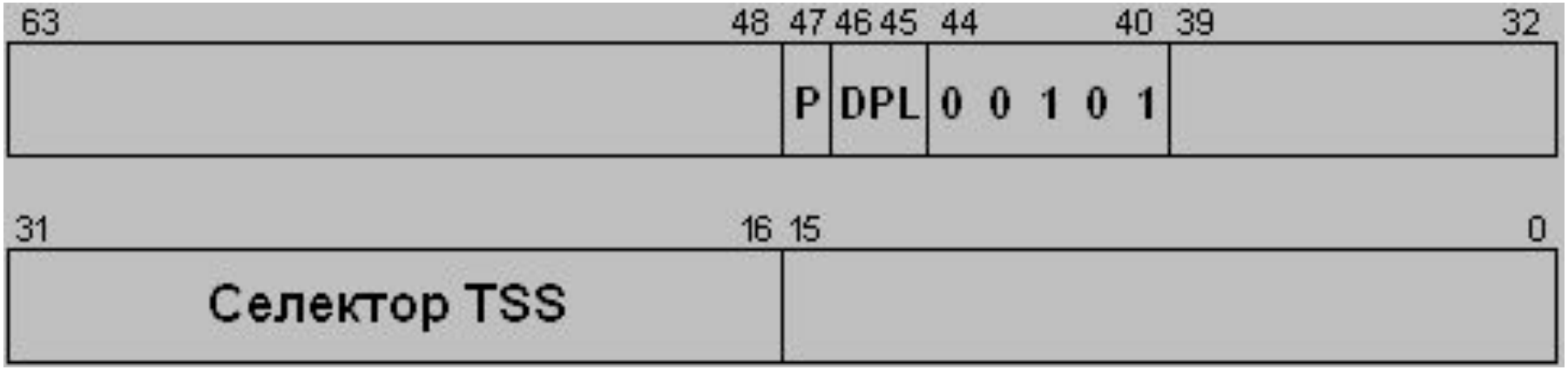


Схема шлюза задачи.

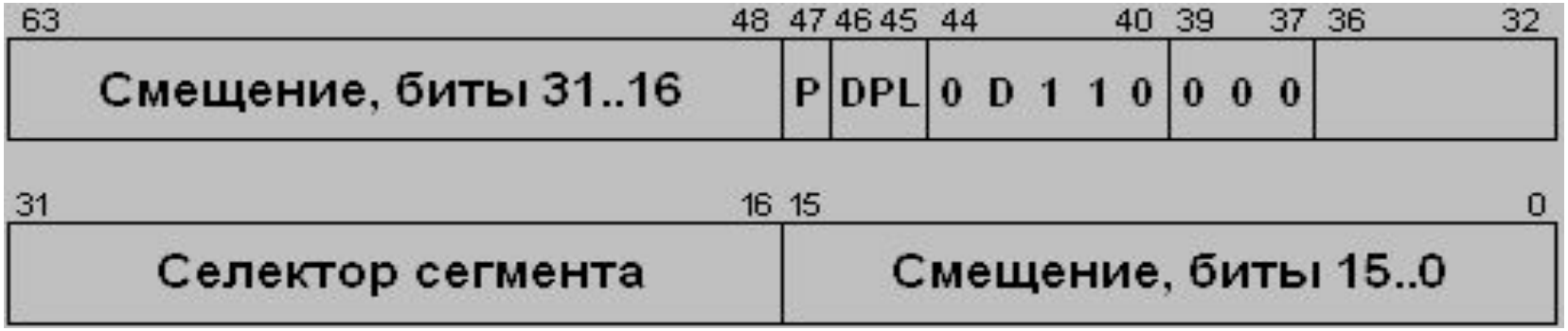
Обратите внимание на то, что бит 4 в access_rights, соответствующий биту S в формате дескриптора, равен 0. Это значит, что дескриптор описывает системный объект и биты 0..3 в access_rights определяют тип этого объекта.

Первое и последнее слова (dw) в формате дескриптора содержат 0, т. к. любая задача определяется своим дескриптором, на который и ссылается селектор TSS (слово состояния задачи)

Шлюз прерывания.

```
dw offset_low ; Младшая часть смещения
 dw selector ; Селектор сегмента кода
db 0
db access_rights ; Права доступа
dw offset_hi ; Старшая часть смещения
```

Шлюз прерывания через селектор и смещение задаёт адрес обработчика прерывания.



Шлюз ловушки.

dw offset_low ; Младшая часть смещения

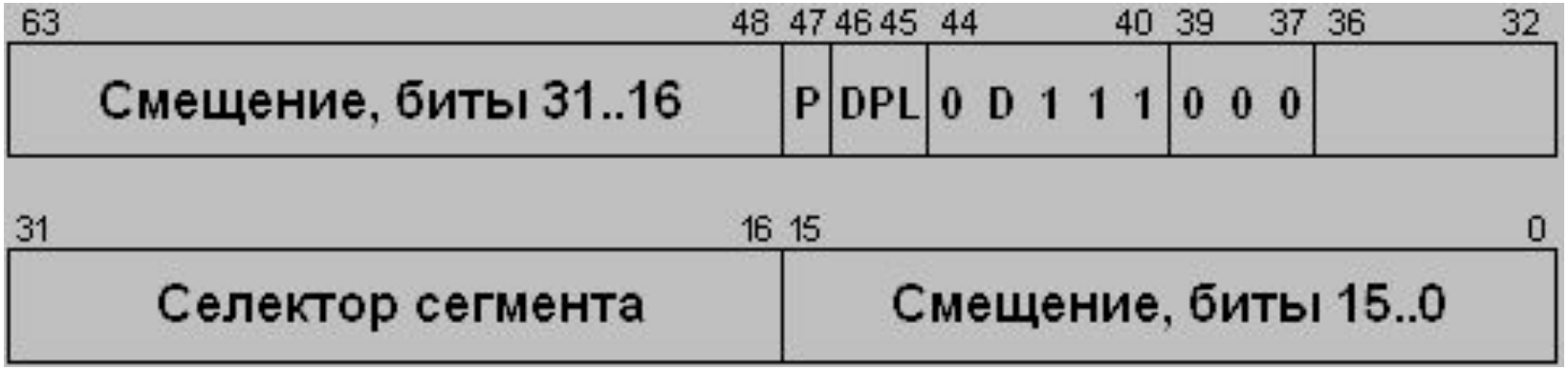
dw selector ; Селектор сегмента кода

db 0

db access_rights ; Права доступа

dw offset_hi ; Старшая часть смещения

Шлюз ловушки через селектор и смещение задаёт адрес обработчика прерывания.



Примечание.

D - это размер шлюза: 1 = 32 бита; 0 = 16 бит. Размер шлюза определяет размер стека, используемый процессором по умолчанию. Перед вызовом обработчика, процессор помещает в стек значения регистров CS, EIP, EFLAGS и иногда SS, ESP и dw-код ошибки. Если размер шлюза - 32 бита, то значения размером в 16 бит будут расширены нулями до 32-х.

Исключения.

Исключениями называются прерывания, которые генерирует процессор в ответ на нарушения условий защиты. Повлиять на исключения прикладные программы (работающие на уровне привилегий, выше 0) не могут, замаскировать - тоже.

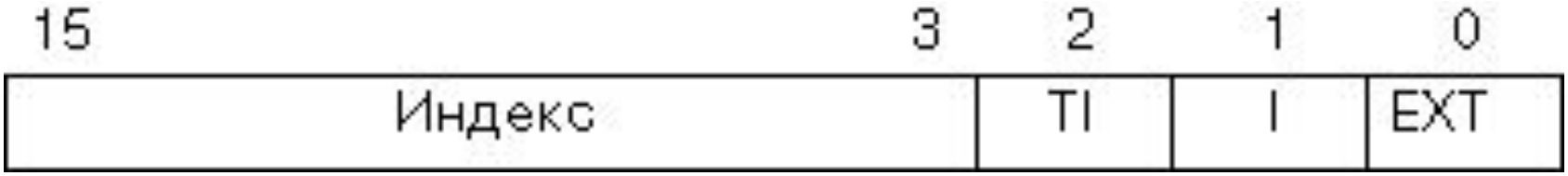
Исключения делятся на три типа, в зависимости от условий их возникновения: 1. Ошибка (fault) 2. Ловушка (trap) 3. Авария (abort)

Ошибка - это исключение, возникающая в ситуации ошибочных действий программы и подразумевается, что такую ошибку можно исправить. Такой тип исключения позволяет рестарт "виноватой" команды после исправления ситуации, для чего в стеке обработчика адрес возврата из прерывания указывает на команду, вызвавшую исключение.

Ловушка - это исключение, возникающее сразу после выполнения "отлавливаемой" команды. Это исключение позволяет продолжить выполнение программы со следующей команды (без рестарта "виноватой"). На ловушках строится механизм отладки программ.

Авария - это исключение, которое не позволяет продолжить выполнение прерванной программы и сигнализирует о серьёзных нарушениях целостности системы. Примером аварии служит исключение двойного нарушения (прерывание 8), когда сама попытка обработки одного исключения вызывает другое исключение.

Перед тем, как передать управление обработчику исключения, для многих зарезервированных прерываний процессор помещает в стек 16-битовый код ошибки. Этот код ошибки программа может проанализировать и тем самым получить некоторую дополнительную информацию об ошибке.
Формат кода ошибки приведён на рис.



Поле индекса содержит индекс дескриптора, при обращении к которому произошла ошибка. Поле I, равное 1, означает, что этот индекс относится к таблице IDT. В этом случае произошла ошибка при обработке прерывания или исключения.
Если бит I равен 0, поле TI выбирает таблицу дескрипторов (GDT или LDT) по аналогии с соответствующим полем селектора.
Бит EXT устанавливается в том случае, когда ошибка произошла не в результате выполнения текущей команды, а по внешним относительно выполняемой программы причинам.

Программа сможет проанализировать этот код только для следующих исключений:
08h - двойная ошибка;
0Ah - недействительный TSS;
0Bh - отсутствие сегмента в памяти;
0Ch - исключение при работе со стеком;
0Dh - исключение по защите памяти.
Заметим, что аналога коду ошибки для зарезервированных прерываний в реальном режиме нет.
Кроме того, новым при обработке прерываний в защищённом режиме является свойство повторной запускаемости исключений. Свойством повторной запускаемости обладают не все исключения.

Далее приводится полный список исключений и прерываний. В этой таблице применяются следующие обозначения:

Номер вектора - номер вектора прерывания, на которое отображено исключение.

Название - используется в документации Intel и состоит из заглавных букв английского названия исключения, например, #DE - Divide Error.

Error code - наличие dw-кода ошибки, который процессор добавляет в стек обработчика перед передачей ему управления.

Номер вектора	Название	Описание	Тип	Error Code	Источник исключения
0	#DE	Ошибка деления	Fault	Нет	Команды DIV и IDIV
1	#DB	Отладка	Fault/Trap	Нет	Любая команда или команда INT 1
2	-	Прерывание NMI	Прерывание	Нет	Немаскируемое внешнее прерывание
3	#BP	Breakpoint	Trap	Нет	Команда INT 3
4	#OF	Переполнение	Trap	Нет	Команда INTO
5	#BR	Превышение предела	Fault	Нет	Команда BOUND
6	#UD	Недопустимая команда (Invalid Opcode)	Fault	Нет	Недопустимая команда или команда UD2[1]
7	#NM	Устройство не доступно (No Math Coprocessor)	Fault	Нет	Команды плавающей точки или команда WAIT/FWAIT
8	#DF	Двойная ошибка	Abort	Да (Нуль)	Любая команда
9	-	Превышение сегмента сопроцессора (зарезервировано)	Fault	Нет	Команды плавающей точки[2]
0Ah	#TS	Недопустимый TSS	Fault	Да	Переключение задач или доступ к TSS

0Bh	#NP	Сегмент не присутствует	Fault	Fault	Загрузка сегментных регистров или доступ к сегментам
0Ch	#SS	Ошибка сегмента стека	Fault	Да	Операции над стеком и загрузка в SS
0Dh	#GP	Общая защита	Fault	Да	Любой доступ к памяти и прочие проверки защиты
0Eh	#PF	Страничное нарушение	Fault	Да	Доступ к памяти
0Fh	-	Зарезервировано Intel-ом. Не использовать.		Нет	
10h	#MF	Ошибка плавающей точки в x87 FPU (Ошибка математики)	Fault	Нет	Команда x87 FPU или команда WAIT/FWAIT
11h	#AC	Проверка выравнивания	Fault	Да	(Нуль) Обращение к пямяти[3]
12h	#MC	Проверка оборудования	Abort	Нет	Наличие кодов и их содержимое зависит от модели[4]
13h	#XF	Исключение плавающей точки в SIMD	Fault	Нет	Команды SSE и SSE2[5]
14h-1Fh	-	Заререзвировано Intel-ом. Не использовать			
20h-FFh	-	Прерывания определяются пользователем	Прерывание		Внешнее прерывание или команда INT n

Некоторое небольшое число исключений, являющиеся ошибками, не позволяют продолжить выполнение программы, т.к. при их генерации теряется часть данных программы, в которой произошла ошибка. Операционная система должна определить дескрипторы для всех исключений. Если процессор при генерации исключения не обнаружит в IDT соответствующего дескриптора, это приведёт к генерации ещё одного исключения. Процессор может обработать два исключения последовательно, но в некоторых случаях, когда это ему не удаётся, он генерирует исключение двойной ошибки.

Если в IDT не будет дескриптора и для исключения двойной ошибки, либо не сможет корректно его обработать, то процессор переходит в режим отключения, похожий на режим, в который его переводит команда HLT (т.е. попросту, зависает) и будет реагировать только на сигналы типа аппаратного сброса.

Сами обработчики исключений не обязательно должны выполнять свои функции. На этапе создания операционной системы достаточно сделать "заглушки", которые будут выводить на экран номер исключения и параметры, переданные в стеке. Т.к. действия обработчиков сильно отличаются в зависимости от предназначения ОС
Все дескрипторы прерываний и исключений объединяются в одну таблицу IDT (Interrupt Desriptor Table). Сама IDT может располагаться в памяти по любому адресу и состоять из любого числа дескрипторов в пределах от 0 до 256. В отличие от GDT, нулевой дескриптор в IDT используется нулевым вектором (исключение деления на 0).

Для неиспользуемых векторов бит P дескрипторов должен быть равен 0, тогда при попытке обращения к нему процессор будет генерировать исключение неприсутствующего сегмента и ОС сможет корректно обработать неиспользуемое прерывание. В противном случае, скорее всего, возникнет другое исключение, тип которого заранее предусмотреть невозможно.

Для повышения производительности системы, рекомендуется размещать IDT по адресу, кратному 8. Размер IDT должен быть кратен 8, т.к. она состоит из 8-байтных дескрипторов, а предел, следовательно, на 1 меньше.

Если произойдёт обращение к вектору прерывания, дескриптор которого должен находиться за пределами IDT, то процессор сгенерирует исключение общей защиты.

Возникло исключение 0Bh - отсутствие сегмента в памяти. Обработчик этого исключения, входящий в состав операционной системы выполнил свопинг соответствующего сегмента в оперативную память. Что дальше? А дальше было бы неплохо повторить выполнение прерванной команды! Это можно сделать, так как для всех повторно запускаемых исключений (кроме 03h - прерывание по точке останова и 04h - переполнение) в стек включается адрес не следующей за прерванной командой, а адрес первого байта команды, которая вызвала исключение. Выполнив команду IRET, программа обработки исключения вновь передаст управление прерванной команде.
Свойством повторной запускаемости обладает большинство зарезервированных прерываний, кроме следующих:
01h - прерывание для пошаговой работы;
08h - двойная ошибка;
09h - превышение сегмента сопроцессором;
0Dh - исключение по защите памяти;
10h - исключение сопроцессора.

Параметры IDT (адрес и предел) процессор хранит с специальном 48-разрядном регистре IDTR. Формат этого регистра следующий:
биты: 0..15: 16-разрядный предел IDT
16..48: 32-разрядный адрес начала IDT
Адрес начала IDT - это тот адрес, по которому вы разместили IDT.
Предел таблицы IDT - это максимальное смещение относительно её начала.

Для загрузки содержимого IDTR из памяти в регистр используется команда LIDT, для сохранения из регистра в память - SIDT, причём, команда IDTR может выполняться только на нулевом уровне привилегий, а SIDT - на любом. Единственным операндом у обеих команд является адрес 48-разрядной переменной.

Программа, работающая не на 0-м уровне привилегий может получить адрес и предел IDT и тут уже от самой операционной системы зависит, разрешит ли она доступ непривилегированной программе к IDT.

Обработка аппаратных прерываний

Вспомните диапазон номеров прерываний, используемый в реальном режиме в компьютерах IBM PC: для обработки прерываний IRQ0-IRQ7 используются номера прерываний от 08h до 0Fh, а для IRQ8-IRQ15 - от 70h до 77h.

Но в защищённом режиме номера от 08h до 0Fh зарезервированы для обработки исключений!

В связи с этим возникает необходимость при установке системы прерываний в защищённом режиме перенаправить аппаратные прерывания на другие вектора, лежащие за пределами 00..1Fh, а при возврате в режим реальных адресов - обратно, на 8..0F и 70h..7Fh. После возврата процессора в реальный режим необходимо восстановить состояния контроллера прерываний.

Обработкой аппаратных прерываних в процессорах Intel386 и Intel486 занимается микросхема 8259A, а в Pentium и старше - APIC (Advanced Programmable Interrupt Controller). APIC обладает замечательным свойством - его можно отключить или, другими словами, запретить (disable) и тогда он будет эмулировать работу внешнего контроллера прерываний 8259A.

Обработка аппаратных прерываний

Вспомните диапазон номеров прерываний, используемый в реальном режиме в компьютерах IBM PC: для обработки прерываний IRQ0-IRQ7 используются номера прерываний от 08h до 0Fh, а для IRQ8-IRQ15 - от 70h до 77h.

Но в защищённом режиме номера от 08h до 0Fh зарезервированы для обработки исключений!

В связи с этим возникает необходимость при установке системы прерываний в защищённом режиме перенаправить аппаратные прерывания на другие вектора, лежащие за пределами 00..1Fh, а при возврате в режим реальных адресов - обратно, на 8..0F и 70h..7Fh. После возврата процессора в реальный режим необходимо восстановить состояния контроллера прерываний.

Обработкой аппаратных прерываних в процессорах Intel386 и Intel486 занимается микросхема 8259A, а в Pentium и старше - APIC (Advanced Programmable Interrupt Controller). APIC обладает замечательным свойством - его можно отключить или, другими словами, запретить (disable) и тогда он будет эмулировать работу внешнего контроллера прерываний 8259A.

```
;Обработка аппаратных прерываний

; Включение локального APIC
enable_APIC proc near mov ecx,1bh
db 0fh, 32h ; Код команды RDMSR
or ah,1000b ; Устанавливаем 11-й бит в MSR 1Bh
db 0fh, 30h ; Код команды WRMSR
ret
endp


;Отключение локального APIC
and ah,11110111b ; Сбрасываем 11-й бит в MSR 1Bh
```

Итак, для того, чтобы определить прерывания в защищённом режиме, нужно выполнить следующие действия:

1. Перенаправить аппаратные прерывания (IRQ)

2. Создать дескрипторы для всех используемых векторов (исключений, аппаратных и программных прерываний).

3. Подготовить образ IDTR и загрузить его в регистр IDTR.

4. Разрешить прерывания

при реализации обработчиков аппаратных прерываний придерживайтесь следующего:

1. Не используйте в IDT шлюзы ловушек, а только прерываний, т.к. при переходе через шлюз прерывания процессор автоматически запрещает маскируемые прерывания (сбрасывая флаг IF в EFLAGS), но не делает этого для шлюза ловушки.

2. В начале обработки прерывания посылайте в контроллер 8259A команду конца прерывания (EOI).

3. Постарайтесь сделать обработку прерывания как можно быстрее, т.к. процессор не допустит генерации нового прерывания, пока не будет завершён обработчик.

4. Как правило, операционная система защищённого режима подразумевает возврат в режим реальных адресов и выход в ту ОС, из которой её запускали (например, в MS-DOS). В таком случае необходимо предусмотреть правильное маскирование прерываний IRQ перед возвратом в такую ОС, так как обычно не все прерывания разрешены.

5. MS-DOS приготовил один неприятный "подводный камень". Дело в том, при условии, что выполняются какие-либо процессы, длительностью более, чем примерно 2 секунды, контроллер клавиатуры генерирует символ. Если не обработать его должным образом, то клавиатура будет заблокирована, поэтому предлагается следующее: 1. Обязательно размаскировывать прерывание клавиатуры (IRQ 1). 2. Обязательно разрешать прерывания на время выполнения части программы, работающей в защищённом режиме. 3. Установить обработчик IRQ клавиатуры или хотя бы следующую заглушку:

```
IRQ_1_handler      macro

      push   ax

      in     al,60h        ; AL содержит скан-код клавиатуры, но в
                           ;  этом примере он не сохраняется -
                           ;  обработчик IRQ 1 работает как заглушка.
      in       al,61H
      mov      ah,al
      or       al,80h
      out      61H,al
      xchg     ah,al
      out      61H,al

      mov    al,20h
      out    20h,al

      pop    ax

      iret

endm
```