Лекция №6. Аппаратные прерывания в защищенном режиме

Переход к защищенного режима и возвращения к реальному.

Перед переходом в защищенный режим программисту нужно сообщить процессору физический адрес таблицы глобальных дескрипторов и ее размер (предел). для этого загрузить в регистр процессору GDTR информацию о таблице глобальных дескрипторов: линейную базовый адрес таблицы и ее границу.

эта информация размещается в первые 6 байтах поля данных (Псевдодескриптором). для загрузки GDTR предусмотрена специальная привилегированная команда lgdt (load global descriptor table), которая требует указания в качества операнда имени псевдодескриптором.

7	6	5	4	3	2	1	0	байты
-		Линейная базовый адрес				предел		описание

Рис. 1 Формат псевдодескриптором

Если таблица глобальных дескрипторов расположена в начале сегмента данных и ее базовый адрес совпадает с базовым адресом всего сегмента, то заполнение псевдодескриптором упрощается.

Запретить все аппаратные прерывания . В защищенном режиме процессор выполняет процедуры прерываний не так, как в реальной. При поступлении сигнала прерывания процессор не обращается к таблице векторов прерываний в первом килобайте памяти, как в реальном режиме, а вытягивает адрес программы обработки прерывание из таблицы дескрипторов прерываний.

Перевод процессора в защищенный режим можно выполнить командами SMSW (Store machine status word, сохранения слова состояния машины) и LMSW (load machine status word). Нужно установить бит 0 слова состояния машины (регистра CR0) Это бит PE - бит разрешения защиты.

Поскольку другие биты этого слова нам не известны, нужно прочитать в регистр AX слово состояния машины, установить в нем бит 0 и записать модифицированное слово состояния назад, в процессор.

Все последующие команды будут выполняться в защищенном режиме.

Программист имеет дело с селекторами, т.е. номерами дескрипторов, а процессор - с самими дескрипторами, хранящихся в теневых регистрах , какие существуют для каждого из сегментных регистров.

Теневые регистры недоступны программисту, они автоматически загружаются процессором из таблицы дескрипторов каждый раз, когда процессор загружает соответствующий сегментный регистр.

Именно содержание теневого регистра (в первую очередь, линейный адрес сегмента) определяет область памяти, к которой обращается процессор при выполнении конкретной команды. После перехода в защищенный режим нужно загрузить в сегментные регистры селекторы соответствующих сегментов. Это позволит процессору правильно заполнить все поля теневых регистров из таблицы дескрипторов. Пока эта операция не выполнена, некоторые поля теневых регистров (границы сегментов) могут содержать неверную информацию.

В современных процессорах с целью повышения скорости выполнения программ используется конвейерная обработка команд. Одновременно с выполнением текущей команды осуществляется выборка операндов следующей, дешифрации третьей и выборка из памяти четвертой. Итак, в момент перехода в защищенный режим уже могут быть расшифрованы несколько следующих команд и выбраны из памяти их операнды. Но эти действия выполнялись по правилам реального режима. Поэтому нужно очистить очередь передвиборкы перед загрузкой селектора в регистр CS (команда дальнего перехода).

Но, если программа начинала работу под управлением системы, то и закончить ее так же нужно обычным способом, чтобы не повредить работоспособности системы. В защищенном режиме нельзя обращаться к функциям системы и BIOS, потому что это программы реального режима. В них используется загрузки в сегментные регистры сегментных адресов.

В защищенном режиме в эти регистры загружаются селекторы. Кроме того, использование в защищенном режиме команд с определенными номерами, приведет к других результатов. То есть программу, работающую в защищенном режиме, нельзя завершать средствами системы. Сначала ее нужно вернуть в реальный режим.

Возвращение в реальный режим можно осуществить сбросом процессора. действия процессора после сброса определяются одной из ячеек КМОП-микросхемы -

байтом состояния отключения, расположенным по адресу Fh. Если в этом байте записан код Ah, после сброса процессору управления передается по адресу, извлекается из двосливнои ячейки 40h: 67h, расположенной в области данных BIOS.

Таким образом, для подготовки возвращения в реальный режим мы должны в ячейку 40h: 67h записать адрес возврата, а в байт Fh КМОП-микросхемы занести код Ah. Точка возврата может располагаться в любом месте программы.

Другой метод перехода в защищенный режим заключается в открытии линии A20. это адресная линия, на которой устанавливается единичный уровень сигнала, если происходит обращение к мегабайт адресного пространства с номерами 1,3,5 и т.д. (первый мегабайт имеет номер 0).

В реальном режиме линия A20 заблокирована и если значение адреса выходит за пределы FFFFFh, выполняется его циклическое вращение (линейный адрес 1.000000h превращается в 00000h и т.д.).

Открытие линии A20 выключает механизм циклического вращения адресов. управление блокировкой линии A20 осуществляется через порт 64h, куда сначала нужно отправить команду D1h управления линией A20, затем - код открытия (DFh).

Перед переходом в реальный режим нужно закрыть линию A20. Для этого в порт 64h засылается команда D1h и затем - DDh (закрытие линии).

Переход в реальный режим.

Сброс процессора осуществляется засылки команды FFh в порт 64h контроллеру клавиатуры. Эта команда возбуждает сигнал на одном из выводов контроллеру клавиатуры, что приводит к появлению сигнала сброса на выводе RESET микропроцессора.

Перед выполнением сброса нужно сохранить текущее содержимое SP, чтобы после перехода в реальный режим можно было восстановить состояние стека.

После сброса процессор работает в реальном режиме. управление передается программам BIOS. BIOS анализирует состояние байта отключения КМОП-микросхемы и осуществляет переход по адресу, хранящемуся в ячейке 40h: 67h области данных BIOS.

Сброс процессора выполняется не мгновенно. Поэтому, чтобы процессор не выполнил до своей остановки еще несколько команд, нужно организовать ожидания сброса процессора (команды hlt или бесконечного цикла). Далее нужно позволить прерывания.

Внутренние прерывания процессора в защищенном режиме.

Исключение.

Обработка прерываний в защищенном режиме очень отличается от обработки прерываний в реальном режиме. Если перейти в защищенный режим, не выполнив запрета аппаратных прерываний, первое же такое прерывание (например, от таймера) привело бы к отключению процессора.

В реальном режиме 32-разрядных процессоров предусмотрены прерывания 3-х видов:

- внутренние , Возникающие в самом МП;
- внешние , Поступающих в процессор от внешних устройств компьютера через контроллеры прерываний;
- программные , Инициированные командой int.

В защищенном режиме возможны все эти типы прерываний, но функции внутренних прерываний и их количество существенно расширены. Внутренние прерывания называются здесь исключениями, исключительными ситуациями или особыми случаями (exception).

Все прерывания защищенного режима, как и реального, имеют свои номера, причем их общее количество не должно превышать 256. В исключения отданы первые 32 номера 0 ... 31. Реально возникающие исключения имеют номера 0 ... 17 а номера 18 ... 31 зарезервированы для будущих моделей процессоров.

В защищенном режиме аналогом таблицы векторов прерываний является таблица дескрипторов прерываний (IDT - Interrupt Descriptor Table), которая располагается в операционной системе или в программе пользователя. Таблица IDT содержит дескрипторы обработчиков прерываний, в которые входят их адреса. Для того, чтобы процессор смог обращаться к этой таблице, ее адрес нужно загрузить в регистр IDTR (командой lidt).

Таблица дескрипторов прерываний состоит из дескрипторов, которые называются шлюзами (вентилями). Через шлюзы осуществляется доступ к обработчиков прерываний и исключений. Процессор, зарегистрировав прерывания или исключения, по его номером извлекает из IDT шлюз, определяет адрес обработчика и передает ему управление.

Обработчик должен заканчиваться командой iret, что возвращает управление в прерванную программу. То есть, в программе, которая обслуживает исключения, надо:

1. Сформировать таблицу дескрипторов прерываний IDT;

2. Поместить в нее адреса обработчиков исключений, предусмотренных в программе;

3. Скачать адрес IDT в системный регистр процессору IDTR;

4. Перейти в защищенный режим.

Пока процессор находится в защищенном режиме, он при возникновении прерываний будет использовать IDT. По возвращении в реальный режим (до разрешения прерываний) ее надо заменить таблицей векторов реального режима.

В таблицу IDT могут входить шлюзы следующих типов:

1. Шлюз вызова МП286 (тип 4);

2. Шлюз задачи (тип 5);

3. Шлюз прерывания МП286 (тип 6);

4. Шлюз ловушки МП286 (тип 7);

5. Шлюз вызова МП386, 486 и Pentium (тип Ch)

6. Шлюз прерывания МП386, 486 и Pentium (тип Eh)

7. Шлюз ловушки МП386, 486 и Pentium (тип Fh).

через шлюз задачи (task) осуществляется переключение на другие задачи в многозадачном режиме.

через шлюзы прерываний (interrupt) обслуживаются аппаратные прерывания.

через шлюзы ловушек (trap) осуществляется обработка исключений и программных прерываний.

Формат шлюзов.

Рассмотрим формат дескрипторов (рис. 2), из которых строится таблица прерываний IDT (шлюзов). В IDT могут входить шлюзы трех типов: ловушек, прерываний и задач.

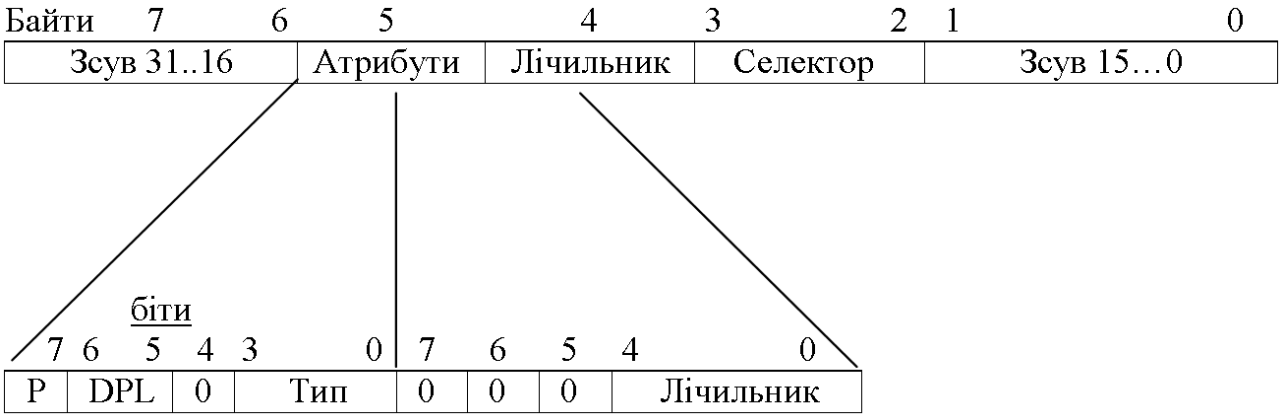

Рис. 2. Формат шлюзов

Если основной частью содержания сегмента памяти была его линейный адрес, то в шлюзе указывается полная трьохсливна адрес обработчика, состоящий из селектора и смещения.

Смещение имеет 32 бита и занимает в шлюзе 2 поля - байты 0-2 и 6-7. Селектор имеет 16 бит и занимает байты 2,3. Байт 4 не используется.

Поле счетчика предназначенное для хранения числа параметров, которые копируются с одного стека на другой в тех случаях, когда осуществляется переход с одного уровня привілеів на другой. Каждый параметр - это двойное слово.

Байт атрибутов имеет такую же структуру, как и в дескрипторах памяти и включает тип, идентификатор дескриптору (бит 4), уровень привилегий дескриптору DPL и бит присутствии P. Тип дескриптору может принимать 16 значений, но в IDT допустим описывать только шлюзы задач, прерываний и ловушек.

Значение поля типа для системных дескрипторов и шлюзов:

- 0 Не определено;
- 1 Свободный сегмент состояния задачи TSS 80286;
- 2 LDT;
- 3 Занят сегмент состояния задачи TSS 80286;
- 4 Шлюз вызова 80286;
- 5 Шлюз задачи;
- 6 Шлюз прерывания 80286;
- 7 Шлюз ловушки 80286;
- 8 Не определено;

- 9 Свободный сегмент состояния задачи;
- Ah Не определено;
- Bh Занят сегмент состояния задачи TSS 386, 486, Pentium;
- Ch Шлюз вызова 80386, и486, Pentium;
- Dh Не определено;
- Eh Шлюз прерываний 80386, и486, Pentium;
- Fh Шлюз ловушки 80386, и486, Pentium.

Обработка исключений.

При возникновении исключения процессор умножает его номер на 8 и полученный произведение использует, как индекс в таблице дескрипторов прерываний IDT. первые 18 дескрипторов IDT должны всегда описывать программы обработчиков исключений.

Исключение подразделяются на три класса: нарушения, ловушки и аварии.

Нарушение или отказ (fault) - это исключение, которое фиксируется еще до выполнения команды или в процессе ее выполнения. Примеры нарушений: адресация по неустановленным чертой сегмента или обращения к отсутствующему дескриптору. при обработке нарушения процессор оставляет в стеке адрес той команды, выполнение которой привело к исключению. При этом предполагается, что в обработчике нарушения его причина будет ликвидирована, после чего команда iret вернет управления на ту же же, еще невыполненную команду. То есть, сам механизм обработки нарушений предполагает восстановление этого программного сбоя.

Ловушка (trap) обрабатывается процессором после выполнения команды, вызвавшей это исключение и в стеке сохраняется адрес не этой, а следующей команды. Есть, после возвращения из обработчика ловушки исполняется не команда, инициировала исключения, а следующая команда программы. К ловушек относятся все команды программных прерываний int.

Авария или выход из процесса (abort) является следствием серьезных ошибок которые не восстанавливаются, например, обнаружение в системных таблицах неразрешенных или несовместимых значений. Адрес, хранящийся в стеке, не позволяет локализовать команду, которая вызвала исключение, и восстановление программы не предвидится. Обычно аварии требуют перезагрузки системы.

Процессор, зарегистрировав то или иное исключения, сохраняет в стеке содержимое расширенного регистра флагов EFLAGS, селектор сегмента команд, смещение точки возвращения и, в некоторых случаях, 32-битный код ошибки.

Состояние стека при выключении.

Даже если программа выполняется в режиме 16-битных адресов и операндов, как смещение возврата выступает 32-битный содержимое указателя команд EIP, в котором старшая половина равна нулю.

Двосливни данные располагаются в стеке в следующем порядке: в слове стека с меньше адресу - младшая часть 32-битного данного, а в слове стека с большей на 2 адресу - старшая часть.

Для выравнивания стека под содержание CS также выделяется двойное слово в младшей половине которого располагается CS, а старшая половина ничем не заполняется. В режиме 16-разрядных операндов используется только младшая половина ESP (SP).

Перед завершением обработчика (командой iret) код ошибки нужно снять с стеке. То есть, для правильной обработки исключения необходимо знать причину его возникновения и вид, а также куда вернется управления после завершения обработчика, и включен в стек код ошибки.

При переходе на обработчик исключения процессор заносит в стек адрес возвращения. В обработчике исключения этот адрес можно извлечь и вывести на экран с номером исключения, возникло.

При извлечении из стека адреса возврата следует учитывать возможность наличии в стеке кода ошибки.

При отладке программ защищенного режима чаще всего возникает исключения 13 нарушения общей защиты.

Например, обращение к сегменту данных по относительному адресу, выходит за его пределы:

mov AX, DS: [data_size - 1]; возникновения исключения 13.

Байт с номером data_size - 1 - последний байт сегмента данных. если бы мы обратились по этому адресу командой чтения байта, все было бы нормально:

mov AL, DS: [data_size - 1]

При чтении целого слова второй байт этого слова находится за пределами сегмента данных и возникает нарушение общей защиты.

Табл. 2 Исключение процессора

вектор	Название исключения	класс исключения	Код ум.	Команды, которые вызывают исключения
0	Ошибка деления	нарушение	Нет	div, idiv
1	исключение налаживание	нарушение	Нет	Любая команда
2	немаскируемое прерывание	авария	Нет	
3	Int3	ловушка	Нет	int 3
4	переполнение	ловушка	Нет	Into
5	нарушение границ массива	нарушение	Нет	Bound
6	Недопустимый код команды	нарушение	Нет	Любая команда
7	сопроцессор недоступен	нарушение	Нет	esc, wait
8	двойное нарушение	авария	так	Любая команда
9	Выход сопроцессора с сегмента	авария	Нет	Команда спивпр. обращение к памяти
10	Недопустимый сегмент состояния задачи TSS	нарушение	так	Jmp, call, iret, прерывания
11	отсутствие сегмента	нарушение	так	команда загрузки сегм. регистра
12	Ошибка обращения к стеке	нарушение	так	Команда обращение к стеке
13	Общая защита	нарушение	так	Команда обращение к памяти
14	страничное нарушение	нарушение	так	Команда обращение к памяти
15	зарезервировано			
16	Ошибка пение процесс.	нарушение	нет	esc, wait
17	Ошибка выравнивание	нарушение	так	Команда обращение к памяти
18 ... 31	зарезервированы			
32 ... 255	Предоставлены пользователю для аппаратных прерываний и команд int.			

Таким образом в защищенном режиме процессор следит за тем, чтобы программа НЕ обращалась к невыделенных ей участков памяти. команда mov AL, CS [0]; нарушение общей защиты должен прочитать в регистр AL первый байт сегмента команд. Но, если атрибут сегмента команд равен 98h, команда выполнена не будет, а пробудит нарушение общей защиты.

Таким образом, в защищенном режиме коды команд в сегменте команд с кодом 98h нельзя ни модифицировать, ни даже читать. (С кодом сегмента 9Ah это делать нельзя).

Нарушение общей защиты возникает в том случае, когда программа обращается к памяти за пределами таблицы дескрипторов.

Допустим, в нашей программе описана таблица, содержащая 5 дескрипторов. Строки, загружают в сегментный регистр селектор, который отсутствует в таблице глобальных дескрипторов, вызывают нарушение общей защиты.

mov AX, 40

mov FS, AX; нарушение общей защиты.

Вторая команда этого фрагмента должна загрузить в теневой регистр, связан с сегментным регистром FS, дескриптор, соответствующий селектору 40, то есть шестой по счету дескриптор.

Такая же ситуация возникает, если в программе встречается команда прерывания INT с номером, отсутствующим в таблице дескрипторов прерываний.

Исключение нарушения стека с №Ch (12) возникает, например, при попытке извлечь слово из пустого стека.

Исключение с вектором 11 возникает при обращении к сегменту, обозначенного как отсутствует. В дескрипторе любого сегмента является бит присутствия сегмента в памяти. Он находится в разряде 7 байта атрибутов 1 и обозначается P от Present.

Бит присутствия предназначен для организации виртуального режима, в котором суммарный размер всех выполняемых одновременно программ может превышать фактический объем оперативной памяти.

В этом случае операционная система сохраняет часть сегментов программ на диске, загружая их в память по мере необходимости на место тех сегментов, к которых временно не происходит обращений. Работа с битом присутствия - функция операционной системы.

Выгрузив какой-то сегмент на диск, система сбрасывает бит P в дескрипторе этого сегмента. Дескриптор находится в памяти. Если программа пытается обращения к отсутствующему сегмента, возникает исключения, заставляет систему загружать необходимый сегмент в память. В его дескрипторе устанавливается бит P, обозначая его присутствующим.

Если приложение берет на себя часть функций операционной системы, она сама может устанавливать и сбрасывать бит присутствия.

Исключение 10, 11, 12, 13 отличаются от других тем, что перед передачей управления обработчику, процессор заносит в стек кроме регистра флагов и адреса возвращение еще и код ошибки (на верхушке стека). Код ошибки имеет формат изображен на рисунке 3.5.

<table>
<tr><td>31 ... 16</td><td colspan="2">15 ... 4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>мусор</td><td colspan="3">индекс дескриптору</td><td>TI</td><td colspan="2">IDT EXT</td></tr>
</table>

Рис. 3 Формат кода ошибки

бит 0 - EXT (External, внешний) - равен 1, если исключение возникло в результате обработки другого исключения или внешнего прерывания.

бит 1 (IDT) = 1, если исключение возникло в результате чтения элемента в таблицы дескрипторов прерываний, может иметь место только при обработке исключения или прерывания.

бит 2 (TI - Table Indicator, индикатор таблицы) равен 1, если дескриптор находится в таблице локальных дескрипторов и 0, если дескриптор глобальный.

Использовав информацию, находящуюся в двосливному коде ошибки, программа может восстановить исходное содержание регистров команды, в процессе выполнения которой возникло исключение. Установив бит присутствия в нужном дескрипторе, нужно выполнить эту же команду.

Аппаратные прерывания в защищенном режиме.

Обработка аппаратных прерываний в защищенном режиме.

Аппаратным прерыванием нужно назначать векторы прерываний отличные от 0 ... 31. В машинах типа IBM PC контроллеры прерываний в процессе загрузки всегда программируются так, что базовый вектор ведущего контроллера равна 8, а подрядного - 70h. Итак, перед переходом в защищенный режим нужно перепрограммировать контроллеры прерываний, назначив ведущему контроллеру, например, базовый вектор 20h = 32.

Очевидно, что перед возвращением в реальный режим контроллеры надо снова перепрограммировать, иначе не смогут работать обработчики аппаратных прерываний BIOS. Но можно поступить иначе: переход в реальный режим организовать по посредством отключения процессору, предварительно загрузив в область данных BIOS по адресу 40h: 67h двословну адрес возврата в программу. Если при этом сослать в байт 0Fh КМОП - микросхемы код 05h, то после сброса процессору BIOS перепрограммирует контроллеры прерываний и передаст управление в указанную точку.

Вторая особенность обработки прерываний связана с форматом дескриптору прерывания (шлюза). В дескрипторе тип шлюза указывается в байте атрибутов 1.

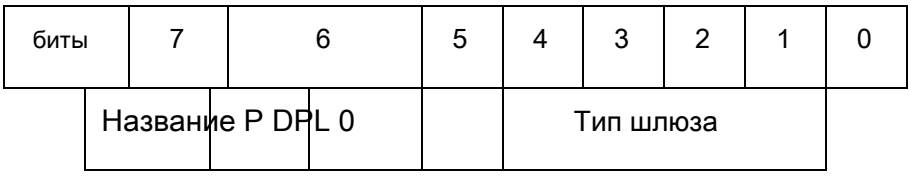

Рис. 4. Формат байта Атрибутов 1:

Поле типа шлюза может принимать 16 различных значений, но в таблице IDT можно описывать только 5, ранее рассмотренных:

1. задачи -5;
2. Тип перерыв. 286 - 6;
3. Ловушка - 7;
4. Тип перив. 386 Eh;
5. Ловушка Fh.

Шлюз задачи используется в тех случаях, когда обработчик прерываний расположен в другой задачи. Шлюзы прерываний и ловушек предполагают наличие обработчиков в текущей задачи.

Поле DPL определяет уровень привилегий шлюза. Это поле проверяется процессором только при выполнении команд программных прерываний - ловушек int и into, чтобы предотвратить обращению пользовательских к системным программных прерываний. Для других исключений и прерываний поле DPL процессором игнорируется.

Бит P в дескрипторе шлюза должен быть установлен в 1. В противном случае, обращения к такому шлюза вызывает исключение.

Таким образом, атрибут дескриптору шлюза типа прерывания должен равняться 8Eh, в отличие от дескриптору ловушек - 8Fh.

Если переход на обработчик осуществляется через шлюз прерывания, процессор сбрасывает при входе в обработчик флаг IF в регистре флагов EFLAGS. Команда iret, загружая из стека сохранен там содержимое EFLAGS, снова разрешает прерывание. При переходе на обработчик через шлюз ловушки состояние флага IF не меняется.