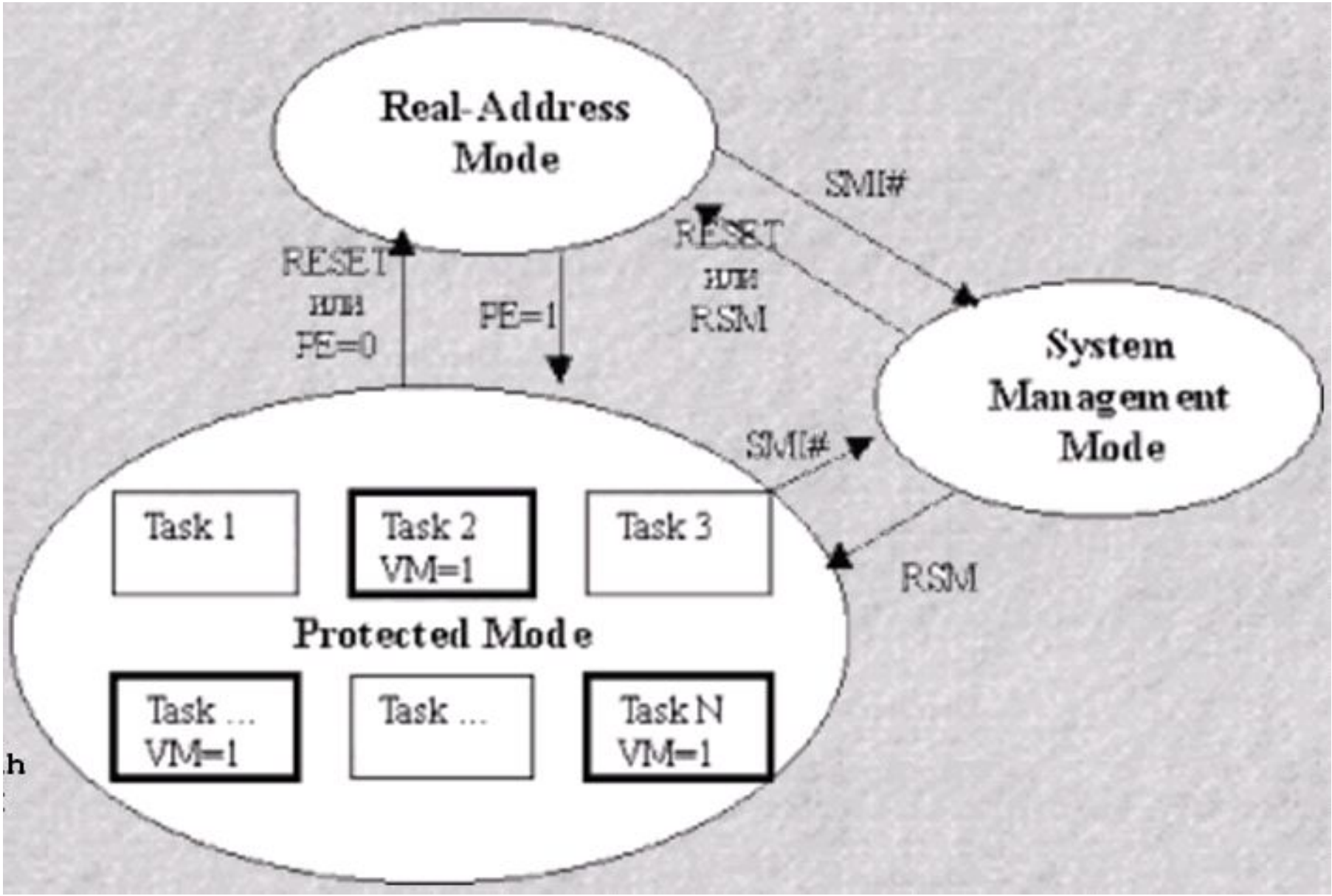
После инициализации процессор находится в режиме реального адреса. Процессор может быть переведен в защищенный режим установкой бита 0 (Protect Enable) в регистре CR0.

Второй вариант "достался в наследство" от 16-тиразрядной архитектуры 80286, для совместимости с которой ее регистр MSW (Machine Status Word) отображается на младшее слово регистра CR0.

Вернуться в режим реального адреса процессор может по сигналу RESET или (в отличие от 80286) сбросив бит PE.

Для совместимости с 80286 инструкция **LMSW** бит PE не сбрасывает.

Режим системного управления изолирован от других режимов. Процессор переходит в этот режим только аппаратно: по низкому уровню на контакте *SMI#* или (для Pentium и выше) по команде с шины APIC. Никакой программный способ не предусмотрен для перехода в этот режим. Процессор возвращается из режима системного управления в тот режим, при работе в котором был получен сигнал *SMI#*. Возврат происходит по команде **RSM**. Эта команда работает только в режиме системного управления и в других режимах не распознается, генерирую исключение #6 (недействительный код операции).


Real-Address Mode
SMI#
RESET
HLT
RSM
RESET
HLT
PE=0
PE=1
System Management Mode
SMI#
RSM
Task 1
Task 2
VM=1
Task 3
Protected Mode
Task ...
VM=1
Task ...
Task N
VM=1

Основы переключения реального и защищенного режимов

Исполнение программ защищенного режима должно начинаться в реальном режиме процессора, то есть в среде «чистого» MS-DOS (без менеджеров оперативной памяти, таких как Himem, EMM386, QEMM и т.п.) или в режиме «Safe mode command prompt only» операционных систем Windows 9x. В операционной системе Windows 2000 (и ей подобных – Win NT/XP) – работа этих программ, по определению, не возможна.

Действия, необходимые для обеспечения функционирования программы защищенного режима :

1. **Подготовка в оперативной памяти таблицы глобальных дескрипторов GDT.**
2. **Инициализация необходимых дескрипторов в таблице GDT.**
3. **Загрузка в регистр gdtr адреса и размера таблицы GDT.**
4. **Запрет обработки аппаратных прерываний.**
5. **Переключение микропроцессора в защищенный режим.**
6. **Организация работы в защищенном режиме:**
   - **настроить сегментные регистры;**
   - **выполнить собственно содержательную работу программы; подготовиться к возврату в реальный режим;**
   - **запретить аппаратные прерывания;**
7. **Переключение микропроцессора в реальный режим.**
8. **Настройка сегментных регистров для работы в реальном режиме.**
9. **Разрешение прерываний и стандартное для MS-DOS завершение работы программы.**

Подготовка таблиц глобальных дескрипторов GDT

Для простейшей программы достаточно определить пока только одну дескрипторную таблицу — глобальную дескрипторную таблицу GDT.

Таблицу LDT есть смысл применять, когда в системе работают несколько задач и необходимо изолировать их друг от друга.

Определимся теперь с набором дескрипторов в таблице GDT, которые понадобятся для нашей программы:

дескриптор для описания сегмента самой таблицы GDT;

дескриптор для описания сегмента данных программы;

дескриптор для описания сегмента команд программы;

дескриптор для описания сегмента стека программы;

дескриптор для описания сегмента, в котором будет находиться процедура для выполнения алгоритмов программы

Загрузка регистра gdtr

Для загрузки регистра gdtr существует команда lgdt:

lgdt адрес_48-битного_поля (Load GDT register) - загрузить регистр gdtr.

Команда lgdt загружает системный регистр gdtr содержимым 6-байтового поля, адрес которого указан в качестве операнда. Вначале необходимо сформировать поле из шести байт со структурой, аналогичной формату регистра gdtr, а затем указать адрес этого поля в качестве операнда команды lgdt.

Для резервирования поля из шести байт (48 бит) TASM поддерживает специальные директивы резервирования и инициализации данных — dp и df. После выделения — с помощью одной из этих директив — области памяти в сегменте данных необходимо сформировать в этой области указатель на начало таблицы GDT и ее размер.

Запрет обработки аппаратных прерываний
без соответствующей настройки первое же прерывание от таймера, которое происходит 18,2 раза в секунду, «подвесит» компьютер. Для этого есть несколько способов: прямым программированием контроллера прерываний и командой микропроцессора cli. Можно использовать любой, только не нужно их сочетать.
Переключение микропроцессора в защищенный режим
О том, что микропроцессор находится в защищенном режиме, говорит лишь состояние бита PE в регистре cr0. Установить этот бит можно двумя способами:
1.Непосредственной установкой бита PE в регистре cr0. Состояние этого бита управляет режимами работы микропроцессора: если PE = 0, то микропроцессор работает в реальном режиме, если PE = 1, то микропроцессор работает в защищенном режиме;
2.Использованием функции 89h прерывания 15h BIOS.
Регистр cr0 программно доступен (в реальном режиме и на нулевом кольце в защищенном), поэтому установить бит PE можно, используя обычные команды ассемблера:

```
mov  eax,cr0
or   eax,0001h
mov  cr0,eax
```

Последняя команда mov переводит микропроцессор в защищенный режим.

Работа в защищенном режиме

Настройка сегментных регистров

Как только микропроцессор оказывается в защищенном режиме, первую же команду он пытается выполнить традиционно: по содержимому пары cs:ip определить ее адрес, выбрать ее и т. д. Но содержимое cs должно быть индексом, указывающим на дескриптор сегмента кода в таблице GDT. Но пока это не так, так как в данный момент cs все еще содержит физический адрес параграфа сегмента кода, как этого требуют правила формирования физического адреса в реальном режиме. То же самое происходит и с другими регистрами. Но если содержимое других сегментные регистров можно подкорректировать в программе, то в случае с регистром cs этого сделать нельзя, так как он в защищенном режиме программно недоступен. Нужно «помочь» микропроцессору сориентироваться в этой затруднительной ситуации. Вспомним, что действие команд перехода основано как раз на изменении содержимого регистров cs и ip. Команды ближнего перехода изменяют только содержимое eip\ip, а команды дальнего перехода — оба регистра cs и eip\ip. Воспользуемся этим обстоятельством, вдобавок существует и еще одно свойство команд передачи управления — они сбрасывают конвейер микропроцессора, подготавливая его тем самым к приему команд, которые сформированы уже по правилам защищенного режима. Это же обстоятельство заставляет нас впрямую моделировать команду межсегментного перехода, чтобы поместить правильное значение селектора в сегментный регистр cs.

В микропроцессоре каждому сегментному регистру соответствует свой теневой регистр дескриптора. Этот регистр имеет размер 64 бит и формат дескриптора сегмента. Смена содержимого теневых регистров производится автоматически всякий раз при смене содержимого соответствующего сегментного регистра. Последние наши действия по изменению содержимого сегментных регистров привели к тому, что мы неявно записали в теневые регистры микропроцессора содержимое соответствующих дескрипторов из GDT. Программисту теневые регистры недоступны, с ними работает только микропроцессор. Если есть необходимость изменить их содержимое, то для этого нужно сначала изменить сам дескриптор, а затем загрузить соответствующий ему селектор в нужный сегментный регистр.

Таким образом, в защищенном режиме программист имеет дело с селекторами, т.е. номерами дескрипторов, а процессор - с самими дескрипторами, хранящимися в теневых регистрах.

В реальном режиме теневые регистры заполняются не из таблицы дескрипторов, а непосредственно самим процессором. В частности, процессор заполняет поле базы каждого теневого регистра линейным базовым адресом сегмента, полученным путем умножения па 16 содержимого сегментного регистра, как это и положено в реальном режиме. Поэтому после перехода в защищенный режим в теневых регистрах находятся правильные линейные базовые адреса, и программа будет выполняться правильно, хотя с точки зрения правил адресации защищенного режима содержимое сегментных регистров лишено смысла.

Тем не менее после перехода в защищенный режим прежде всего следует загрузить в используемые 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 (в частности, границы сегментов) содержат неверную информацию.

Загрузить селекторы в сегментные регистры DS, SS и ES не представляет труда. Но как загрузить селектор в регистр CS, в который запрещена прямая запись? Для этого можно воспользоваться искусственно сконструированной командой дальнего перехода, которая, как известно, приводит к смене содержимого и IP, и CS.

Фрагмент
db OEAh ;Код команды far jmp
dw offset continue ;Смещение
dw 16 ;Селектор сегмента команд
выглядящий совершенно нелепо в сегменте команд, как раз и демонстрирует эту методику.
В реальном режиме мы поместили бы во второе слово адреса сегментный адрес сегмента команд, в защищенном же мы записываем в него селектор этого сегмента (число 16).
Команда дальнего перехода, помимо загрузки в CS селектора, выполняет еще одну функцию - она очищает очередь команд в блоке предвыборки команд процессора. Как известно, в современных процессорах с целью повышения скорости выполнения программы используется конвейерная обработка команд программы, позволяющая совместить во времени фазы их обработки. Одновременно с выполнением текущей (первой) команды осуществляется выборка операндов следующей (второй), дешифрация третьей и выборка из памяти четвертой команды. Таким образом, в момент перехода в защищенный режим уже могут быть расшифрованы несколько следующих команд и выбраны из памяти их операнды. Однако эти действия выполнялись, очевидно, по правилам реального, а не защищенного режима, что может привести к нарушениям в работе программы. Команда перехода очищает очередь предвыборки, заставляя процессор заполнить ее заново уже в защищенном режиме.

Выполнение собственно программы защищенного режима
есть существенные ограничения на ее работу. Прежде всего, это связано с тем, что пока мы не используем прерывания. Так что придется использовать прямой доступ к аппаратуре, используя пространство портов ввода-вывода
Подготовка к возврату в реальный режим
Здесь возникает примерно та же проблема с сегментными регистрами, что была при входе в защищенный режим. Мы упоминали уже о теневых регистрах микропроцессора, но не сказали того, что микропроцессор использует их, даже работая в реальном режиме. При этом поля этих регистров заполнены, конечно, в соответствии с требованиями реального режима:
предел должен быть равен 64 К = 0ffffh;
бит G = 0 (значение размера в поле предела) — это значение в байтах;
байт атрибута равен 10010010 = 92h;
базовый адрес значения не имеет.
Следовательно, перед переходом в реальный режим нужно сформировать эти значения в соответствующих дескрипторах и сделать актуальными эти изменения в теневых регистрах, для чего нужно перезагрузить сегментные регистры. После этого можно переходить в реальный режим.

Запрет аппаратных прерываний

Для нашей программы этого делать и не нужно, так как мы их и не разрешали. Но в целом надо иметь в виду, что перед переходом в реальный режим кроме настройки сегментных регистров нужно будет перестраивать и систему прерываний. Для этого на участке программы, производящем соответствующие действия, необходимо запретить аппаратные прерывания.

Переключение микропроцессора в реальный режим

Для этого необходимо сбросить нулевой бит регистра cr0 (только для 386+). Это можно сделать несколькими способами. К примеру:

```
mov   eax,cr0
and   al,0feh
mov   cr0,eax
```

После сброса бита PE микропроцессор снова оказался в реальном режиме.

Настройка сегментных регистров

После перехода в реальный режим опять возникает проблема с содержимым сегментных регистров. Решается она ранее описанным способом.

Разрешение прерываний

Теперь можно разрешить прерывания. Так как в системе прерываний мы ниче го не меняли, то действия наши тоже были простейшими: а) запрещение аппаратных прерываний для того, чтобы на время смены режима работы микропроцессора нас не беспокоило прерывание от таймера, б) последующее разрешение прерываний после возврата в реальный режим.

В защищенном режиме запрещены любые обращения к функциям DOS или BIOS. Причина этого совершенно очевидна - и DOS, и BIOS являются программами реального режима, в которых широко используется сегментная адресация реального режима, т.е. загрузка в сегментные регистры сегментных адресов. В защищенном же режиме в сегментные регистры загружаются не сегментные адреса, а селекторы. Кроме того, обращение к функциям DOS и BIOS осуществляется с помощью команд программного прерывания int с определенными номерами, а в защищенном режиме эти команды приведут к совершенно иным результатам. Поэтому в программе, работающей в защищенном режиме и не имеющей специальных и довольно сложных средств перехода в так называемый режим виртуального 86-го процессора, вывод на экран можно осуществить только прямым программированием видеобуфера. Нельзя также выполнить запись или чтение файла; более того, нельзя даже завершить программу средствами DOS. Сначала се надо вернуть в реальный режим.

Казалось бы, для возврата в реальный режим достаточно сбросить бит 0 этого регистра. Однако дело обстоит не так просто. Для корректного возврата в реальный режим надо выполнить некоторые подготовительные операции

При работе в защищенном режиме в дескрипторах сегментов записаны, среди прочего, их линейные адреса и границы. Процессор при выполнении команды с адресацией к тому или иному сегменту сравнивает полученный им относительный адрес с границей сегмента и, если команда пытается адресоваться за пределами сегмента, формирует прерывание (исключение) нарушения общей защиты. Таким образом, в защищенном режиме программа не может выйти за пределы объявленных ею сегментов, а также не может выполнить действия, запрещенные атрибутами сегмента.
Как уже отмечалось, дескрипторы сегментов хранятся в процессе выполнения программы в теневых регистрах , которые загружаются автоматически при записи в сегментный регистр селектора.
При работе в реальном режиме некоторые поля теневых регистров должны быть заполнены вполне определенным образом. В частности, поле границы любого сегмента должно содержать число FFFFh, а бит дробности сброшен. Если мы просто перейдем в реальный режим сбросом бита 0 в регистре CR0, то в теневых регистрах останутся дескрипторы защищенного режима и при первом же обращении к любому сегменту программы возникнет исключение общей защиты, так как ни один из наших сегментов не имеет границы, равной FFFFh. Поскольку мы не обрабатываем исключения, произойдет либо сброс процессора и перезагрузка компьютера, либо зависание. Таким образом, перед переходом в реальный режим необходимо исправить дескрипторы всех наших сегментов: команд, данных, стека и видеобуфера(если работали с ним)

Теневые регистры, куда, собственно, надо записать значение границы, нам недоступны. Для из модификации придется прибегнуть к окольному маневру: записать в поля границ всех четырех дескрипторов значение FFFFh, а затем повторно загрузить селекторы в сегментные регистры, что приведет к перезаписи содержимого теневых регистров. С сегментным регистром CS так поступить нельзя, поэтому его загрузку придется выполнить, как и ранее, с помощью искусственно сформированной команды дальнего перехода. Настроив все использовавшиеся в программе сегментные регистры, можно сбросить бит 0 в CR0. После перехода в реальный режим нам придется еще раз выполнить команду дальнего перехода, чтобы очистить очередь команд в блоке предвыборки и загрузить в регистр CS вместо хранящегося там селектора обычный сегментный адрес регистра команд.
сегментные адреса,

Если, однако, в программе встретятся команды сохранения и восстановления содержимого сегментных регистров, например
push DS
pop DS
выполнение программы будет нарушено, так как команда pop DS загрузит в DS не сегментный адрес реального режима, а селектор, т.е. число. Это число будет рассматриваться процессором, как сегментный адрес, и дальнейшие обращения к полям данных приведут к адресации физической памяти, лишеной смысла. Даже если в нашей программе нет строк сохранения и восстановления сегментных регистров, они неминуемо встретятся, как только произойдет переход в DOS по команде int 21h, так как диспетчер DOS сохраняет, а затем восстанавливает все регистры задачи, в том числе и сегментные. Поэтому после перехода в реальный режим необходимо загрузить в используемые далее сегментные регистры соответствующие

Работа с прерываниями в защищенном режиме

Допустим, что мы будем обрабатывать одно аппаратное прерывание (от таймера), две разнотипные исключительные ситуации и обычное пользовательское прерывание.

В общем случае для того, чтобы сделать возможным обработку прерываний в защищенном режиме, необходимо выполнить следующие действия:

1. Инициализировать таблицу IDT.
2. Составить процедуры обработчиков прерываний.
3. Запретить аппаратные прерывания.
4. Перепрограммировать контроллер прерываний i8259A.
5. Загрузить регистр IDTR адресом и размером таблицы IDT.
6. Перейти в защищенный режим.
7. Разрешить обработку прерываний.

Далее микропроцессор, находясь в защищенном режиме, может производить обработку необходимых прерываний. По окончании работы для возврата в реальный режим нужно будет все «поставить на свои места», выполнив для этого последовательность следующих действий:

1. Запретить аппаратные прерывания.
2. Выполнить обратное перепрограммирование контроллера прерываний.
3. Перейти в реальный режим работы микропроцессора.
4. Разрешить обработку аппаратных прерываний.

Инициализация таблицы IDT

так же, как и в реальном режиме, все прерывания защищенного режима имеют свои номера от 0 до 255. Отличие их от прерываний реального режима в том, что под цели микропроцессора (исключительные ситуации или просто исключения) отдано гораздо большее количество номеров — первые 32 вектора с номерами 0...31. Из этих 32 векторов реально используются только 0...17, остальные вектора 18...31 пока зарезервированы для будущих разработок. Для того чтобы сформировать таблицу IDT в целом, необходимо определиться с описанием отдельных ее дескрипторов - шлюзов.

Обработчики прерываний

Расположение и порядок следования процедур обработки прерываний в памяти может быть произвольным. Главное, не забывать поместить их адреса в соответствующие дескрипторы. При написании самих программ обработки прерываний мы должны помнить о том, что при возникновении некоторых прерываний в стек вслед за содержимым регистров eip, cs и eflags может записываться еще и код ошибки. Поэтому в общем случае процедура обработки прерывания должна происходить по следующей схеме действий:

1. Снять со стека и проанализировать код ошибки (если он есть).
2. Сохранить в стеке используемые в обработчике регистры микропроцессора.
3. Выполнить необходимые действия, в том числе подготовить возможный рестарт команды, вызвавшей прерывание. Подобный рестарт подразумевает возобновление выполнения прерванной программы, начиная с команды, инициировавшей процесс прерывания.
4. Восстановить сохраненные на шаге 2 регистры;
5. Выполнить команду iret.

Программирование контроллера прерываний i8259A

При загрузке BIOS производит инициализацию контроллеров прерываний (ведущего и ведомого). При этом BIOS назначает им базовые номера: ведущему — 08h, ведомому — 70h. Из этого следует, что если мы разрешим обработку аппаратных прерываний в защищенном режиме, то первое же прерывание от таймера заставит микропроцессор через таблицу IDT обратиться к процедуре ОБРАБОТЧИК ПРЕРЫВАНИЯ ТАЙМЕРА. Что же делать? Ничего не остается, как перепрограммировать ведущий контроллер на другое значение базового вектора. Ведомый контроллер перепрограммировать не обязательно — значение его базового вектора ничем нам не мешает, так как оно находится в области, отведенной под прерывания пользователя.

Загрузка регистра IDTR

Аналогично таблице GDT нужно сообщить микропроцессору, где находится таблица IDT. Это делается с помощью специально выделенного для этой цели регистра idtr. Таким образом, в защищенном режиме любая задача, имея достаточный уровень привилегий, может создать свою среду для обработки прерываний.

Загрузка регистра idtr производится специальной командой lidt, которая имеет следующий формат:

lidt адрес_48-битного_поля (Load IDT register) — загрузить регистр idtr. Команда lidt загружает системный регистр idtr содержимым 6-байтового поля, адрес которого указан в качестве операнда.

После этого можно перейти в защищенный режим и разрешить аппаратные прерывания. Выполнив работу в защищенном режиме, мы готовимся к обратному переходу в реальный режим. Для этого, кроме восстановления вычислительной среды этого режима, необходимо восстановить и соответствующий механизм прерываний.