**БИ’ТЫ**

1. **В каком режиме происходит обращение к селекторам? Почему?**
2. **Что мы могли использовать в своем обработчике клавиатуры?**
3. **Когда вызывается ваш обработчик клавиатуры?**
4. **Когда вызывается ваш обработчик от системного таймера?**
5. **Почему надо перепрограммировать контроллер прерываний?**
6. **Какой дескриптор сегмента описали для подсчета памяти?**
7. **Какой программе выделили 16 мб? (??????)**
8. **Укажите 2 места, где устанавливается 0 уровень привилегий?**

DPL установили в 0. 0 и 1 биты в селекторах (это RPL). ??? (было на семинаре)

В 2х битах атрибута DPL (Descriptor Privilege level) сегмента кода присваиваем 0, и в 2х битах атрибута RPL (Requested Privilege level) индекса GDT присваиваем 0.

1. **Какую программу мы написали в защищенном режиме?**

Программу 0 уровня привилегий, на нулевом уровне привилегий выполняется ядро ОС…это высочайший уровень привилегий.

1. **А какая программа работает на 0 уровне привилегий?**

Ядро.

1. **Что пришлось создать в этой программе?**

Две системные таблицы. (GDT, IDT)

1. **Для чего дескрипторы сегментов в GDT и прерываний в IDT?**

GDT - чтобы получить возможность адресовать память…иначе мы память не сможем адресовать в защищенном режиме.., IDT – чтобы адресовать обработчики прерываний.

1. **Какую память мы считаем?**

?? Память, которая выделена Досбоксу..больше памяти нам не доступно..

…Объем доступного физического адресного пространства

1. **Какими являются таблицы GDT, IDT?**

Системные.

1. **Где заглушки исключений? (для 13 прерывания (нарушения общей защиты) - особенная)**
2. **Покажите 16,32 разрядные сегменты для данных и кода.**
3. **Покажите дескриптор, в которой установлен max limit и бит гранулярности == 1.**
4. **Покажите переход в реальный режим, что там происходит и обязательно ли это должно делаться (в реальном режиме)?**

?? Описываем границу сегмента в дескрипторе сегмента, потом загружаем в теневые регистры. Не обязательно, мы не вышли за границу FFFF (64кб)…находимся в сегментах реального режима и закрываем нашу программу в этом же сегменте.

1. **Digit == 0, это что означает?**

Значит, мы описали сегменты реального режима.

1. **DPL == 0, значит, программа какого уровня привилегий?**

Нулевого.

1. **Вот у тебя селекторы чему равны? 8, 16, 32….кратны чему?**

Восьми…???? значит первые 3 бита == 0 и TI ==0, значит работаем с глобальной таблицей

Мы поставили виртуальную машину Dosbox, она для нас эмулирует голое железо, мы свою программу защищенного режима запускаем на голом железе. Любая ОС запускается на голом железе, в этом ее задача – управлять вычислительной машиной – голым железом! Это железо программно управляемое, но это делают производители железа.

**Вопросы:**

1. **Почему граница сегмента в реальном режиме 64 кбайта**. Потому что у нас 16битная система(или 16 разрядные регистры), размер сегмента не может превышать 64 кбайта.
2. **Какие сегменты были объявлены в GDT?** Сегменты данных, стека, кода, видеобуфера.
3. **Почему мы не используем первые 32 обработчика**. Фирма Microsoft отвела первые 32 дескриптора под исключения. Я сделал аналогично. У меня на них установлены заглушки. Если попущены первые 32, то нужно перепрограммировать ведущий контроллер с 8 на 32 для того, чтобы адресовать вектором прерывания соответствующее аппаратное прерывание.
4. **Как мы пишем на экран?** Напрямую через функцию (или порты?) вводы\вывода.
5. **Чем мы могли пользоваться для обработки прерывания от клавиатуры?** - Портами ввода-вывода.
6. **- К какому типу команд относится команда LGDT?-** к привилегированным командам.
7. **- В каком режиме мы её вызываем?-** В реальном.
8. **- О чем это говорит?** -О том, что в реальном режиме нет никакой защиты.
9. **Как адресуются прерывания в защищенном режиме?** Через созданную мной таблицу дескрипторов прерываний IDT.
10. **что мы делаем в обработчике прерывания таймера?**

инкримент и выводим X.

;прерывание таймера

new\_int08:

push eax

push edi

mov edi, 0B8000h

xor eax, eax

1. **что такое 0B8000h?** - адресс видеобуффера.

**1) 64 мб это какая память?**

- память кот. мы выделили под вирт. машину. (Нам не доступна вся оперативная память.)

**4) Этот сегмент является сегментом чего? Стека, кода, данных?**

- сегмент, по кот. мы считаем память. (стека данных для подсчета памяти) GDTMEMDESCR

5**) В сегменте кода можно выполнять read - write?**

- Можно, **а в стек можно делать r-w?** да

**6)У нас есть 16-битный сегмент или еще какие - то?**

- Есть еще 32 битный. (в формате дескриптора посмотреть)

**7) показать таблицу дескр. прер.**

**8) что делает 13 dup? Выделяет, но как?**

Почему не выделяем подряд 13 дескрипторов, почему так написали?

- Дублируем

**9) могли бы 32 строки не выделять под исключения?** Да

**10) Что тогда нужно обеспечить, если мы отказываемся от этого?**

- перепрограммировать контроллер. Базовый вектор меняется с 8 на 32 и обратно

**11) Для этого что они перепрограммируют? в результате формируемый вектор прерываний является ...**

**12) Чем является базовый вектор прерываний?**

- смещение в табл. дескр. прер.

**15) Какие сегменты объявили, для чего?**

**17) для чего объявили 16 разр. сегмент**

(посмотреть формат дескрипторов)

**18)заглушки ; обработчик исключения общей защиты (13-я строчка)**  ; обработчик всех остальных исключений

**19) «Чему кратно значение селектора?»**

- «8»

**20) «Что такое индекс?»**

- «Смещение в таблице дескрипторов.»

**22) «Как вы считаете объем оперативной памяти?»**

-«У нас есть сегмент, объемом в 4 гигабайта. Считываем значение из ячейки памяти.

Сохраняем его.

Записываем в ячейку свое значение. Считываем.

Если записанное и считанное значения совпадают, то это память.

Возвращаем в ячейку первоначальное значение.

Если значение не совпадают, то пустота. Мы действуем до первого прокола»

**24) «Какие характеристики имеет сегмент «памяти»?»**

- «Бит гранулярности устанавливаем в 1, для того, чтобы Лимит был в 64 килобайтных единицах.

Этот сегмент 32разрядный, для того, чтобы описать все адресное пространство»

**25)«Расскажите о дескрипторах, которые вы описали.»**

-«Я описал 6 сегментов: нулевой дескриптор(по правилу),

сегмент кода, сегмент данных,

сегмент стека,

сегмент видеопамяти, сегмент, описывающий все линейное адресное пространство»

**27) «Если не включить линию А20 и перейти в защищенный режим, то что будет?»**

- «Если не включить линию А20, то 20ый бит всегда будет равен нулю.

Если мы захотим обратиться к адресам, в которых этот бит равен единице,

то мы не сможем получить к ним доступ. Мы получаем «битую память»»

**28) «Почему вы в реальном режиме используете привилегированную команду lgdt?»**

- «Мы можем это делать, т.к. используем директиву .386»

**29) ряд вопросов по коду, такие как**

**«Покажите где вы входите и выходите из защищенного режима»,**

**«Покажите какие сегменты вы описали»,**

**«Что вы делаете до входа и после входа в защищенный режим».**

**30) Что это за программа? Какие у нее особенности? Что в связи с этим вам пришлось объявить?**

- Это программа, которая выполняет функции ядра ОС.

В связи с этим нам пришлось объявить две системный таблицы дескрипторов:

глобальную – для доступа к памяти и таблицу дескрипторов прерываний –

для доступа к обработчикам прерываний.

**31)Почему вдруг ваша программа стала выполнять функции ОС?**

- Потому что наша программа работает в защищенном режиме с нулевым уровнем привилегий.

**32) Есть у вас системные вызовы?**

- Нет.

**33) Если у вас нет системных вызовов, то какие средства вы используете для того, чтобы вводить и выводить символы?**

- Для этого мы используем бесконечный цикл. Для того, чтобы вывести символ мы напрямую

обращаемся к видеопамяти, т.к. у нас нет ДОСовской команды вывода страницы.

**34) Почему существует деление на простые команды и привилегированные?**

- Привилегированные команды могут выполняться только с нулевым уровнем привилегий.

Это нужно для того, чтобы какая-нибудь программа не смогла нарушать работу системы,

выполнив привилегированную команду.

**35) могут также попросить рассказать про структуру дескриптора таблицы IDT**

(В особенности о том, как из дескриптора данной таблицы получить адрес обработчика прерываний.)

**36) 16мб - Что за число?**

16 мб - количество памяти, выделенной виртуальной машине.

**39) Какие прерывания используются?**

- Клавиатуры

- Таймера

**40) Что использую в обработчике прерываний от клавиатуры?**

Порт клавиатуры

**42) Рассказать цепочку действий от возникновения прерываний до его обработки**

- Контроллер прерывания получает сигнал о прерывании и формирует вектор прерывания, который содержит селектор из ТДП.

- Взяв знаение из регистра IDTR значение базового адреса ТДП. В нём по селектору находим

- По этому дискриптору мы находим дескриптор который уже и содержит селектор, смещение и аттрибуты

- По селектору выбираем дискриптор из ТГД, берём оттуда базовый адрес сегмента и прибавляем егок смещению

- Получаем линейный адрес обработчика прерывания

**43) Как и где осуществляется вход/выход в защищенный режим?**

???

**45) Процесс - программа в стадии выполнения. Имеет свою собственную защищенную (другие не имеют доступа) память.**

**46) Что пришлось создать в памяти?**

Две системные таблицы:

ТГД - таблица глобальеых дескрипторов для правильной адресации по памяти

в таблце дскрипторов должно быть описано столько дескр-тов сколько сегментов использует программа.

ТДП - таблица дескрипторов прерываний для корректного вычисления обработчика прерываний

регистр gdtr - регистр таблицы глобальных дескрипторов для хранения линейного базового адреса и границы таблицы глобальных дескрипторов.

idtr - регистр таблица дескрипторов прерываний для хранения линейного базового адреса и границы таблицы дескрипторов прерываний.

по 48бит. от 0 до 15 предел, от 16 до 47 линейный базовый адрес.

**47) ; структура для описания дескрипторов сегментов.**

SDescr STRUC

limit dw 0 ; предел (размер сегмента). граница (биты 0..15)

base\_l dw 0 ; мл. слово физ-го адреса (поле 32-х разрядного адреса). база биты 0..15

base\_m db 0 ; ст. слово физ-го адреса. база биты 1..23

attr1 db 0 ; байт атрибутов 1

attr2 db 0 ; границы (биты 16..19) и атрибуты 2

baseH db 0 ; база биты 24..31

SDescr ENDS

**граница(limit)** сегмента представляет собой номер последнего байта сегмента. поле границы состоит из 20 бит и разбито на две части. младшие 16 бит занимают байты 0 и 1 дескриптора, а старшие 4 бита входят в байт атрибутов 2, занимая в нём биты 0..3. получается что размер сегмента ограничен величиной 1мб. на самом деле это не так. граница может указываться либо в байтах(и тогда размер сегмента 1мб) либо в блоках по 4кб и тогда размер достигает 4гб. в каких единицах задаётся граница определяет старший бит байта атрибутов 2, называемый битом дробности. если он равен 0 то граница в байтах, если единице то в блоках по 4кб.

**база сегмента** 32 бита определяет начальный линейный адрес сегмента в адресном пространстве процесора. линейным называется не сегмент:смещение, а просто номером байта в адресном пр-ве.

**в байте атрибутов1** задаётся ряд характеристик сегмента.. для сегмента команд байт атр1 должен иметь значение 98h, для сегмента данных(или стека) 92h.

**50) какие сегменты и зачем мы объявили?**

у нас есть обязательный нулевой дескриптор, и дескрипторы для сегментов данных, команд, стека и доп-ного сегмента данных, который мы наложим на видеопамять.

**51) особенность реального режима** - ограничение объёма адресуемой оперативной памяти величиной 1мб.

перевод в защ. реж =

1. увеличие адресуемого пространства до 4гб.

2. возможность работать в виртуальном адресном пространстве.

3. организация многозадачного режима с паралельным выполниеем нескольогго программ(процесов)

4. страничная организация памяти, повыщающая уровень защиты задач друг от друга.

в защищённом режиме процесор выполняет процедуру перрывания не так как в реальном. при поступлении сигнала прерывания процессор не обращается к таблице векторов прерываний в первом кб памяти, как в реальном режиме, а извлекает адрес программы обработки прерывания из таблицы дескрипторов прерываний, построенной схоже с ТГД.

**52) в защ. режиме для каждого сегмента** должен быть определён дескриптор - 8байтовое поле. в котором записываются базовый адрес сегмента, его длина и некоторые другие хар-ки. для обращения к требуемому сегменты программист заносит в сегментный регистр не сегментный адрес, а так называемый селектор, в состав которого входит номер(индекс) соответствующего сегмента дескриптора. процессор по этому номеру находит нужный дескриптор, извлекает из него базовый адрес сегмента и прибавляя к нему указанное в конкретной команде смещение(относителньый адрес), формирует адрес ячейки памяти. индекс дескриптора записываются в селектор начиная с бита3, что эквивалентно его умножению на 8.

**53) Привилегированные команды.** тридцати двухразрядные микропроцесоры отличаются расширенным набором команд, часть которых относится к привелегированным. для того чтобы разрешить транслятору обрабатывать эти команды в текс включена директива .386Р

**54) какие действия необходимо выполнить для корректного возвращения в реальный режим - что пишем в теневые регистры и почему?**

После перехода в защ. режим прога не должна работать, т.к. в регстре CS ещё нет селектора сегмента команд и процессор не может обращаться к этому сегменту. в действительности это не совсем так. в процессоре для каждого из сегментных регистров имеется так называемый теневой регистр дескриптор, который имеет формат дескриптора. теневые регистры недоступны программисту. они автоматически загружаются процессором из таблицы дескрипторов каждый раз, когда процессор инициализирует соответствующий сегментный регистр. таким образом в защ. режиме пр-мист имеет дело с селекторами, т.е. номерами дескрипторов, а процессор с самими дескрипторами, хранящимися в теневых регистрах. именно содержимое теневого регистра(в первую очередь линейный адрес сегмента) определяет область памяти, к которой обращается процесор при выполнении конкретной команды. после перехода в защ. режим прежде всего следует загрузить в используемые сегментные регистры селекторы соответствующих сегментов. это позволит процесору правильно заполнить все поля теневых регистров из таблиц дескрипторов. к регистру CS недопустимо прямое программное обращение, поэтому юзаем переход.

При работе в реальном режиме некоторые поля теневых регистров должны быть заполнены определённым образом. граница ffffh, бит дробности 0, доступ для зписи разрешен. границы всех сегментов должны быть точно равны ffffh.перед переходом в реальный режим необходимо исправить все дескрипторы всех наших сегментов.

линия а20 для обращения к расширенной памяти.

- в идт есть селектор и смещение, селектор нужен, чтобы из гдт выбрать правильный адрес

зачем мы пропускаем 1-е 32 значения в idt - т.к. в МС эти 1-е 32 записи отведено под исключения,

**мы можем так не делать?** да, можем. а мы делали также по образу МС у нас - заглушки. для этого мы перепрограмм ведущий контроллер прер на 32, а если нет - мы должн были перепрогр его на 8

какие сегменты описали и для чего мы описали сегмент кода, стека и данных, видеобуфера

(полные его хар-ки почему они такие) модель flat-памяти. всего один единый сегмент для работы со всей памятью. у него база установлена в 0, лимит максимум (4 гб)

в соответтсвии с выбранной

**ТДглоб какие последствия для кода эта структура имела, что на что написали?** мы можем работать со всей памятью как с одним единым непрерывным массивом данных

полная адресация прерываний в защищенном режиме начиная от котроллера прерываний, заканчивая ТГД

\*\*\* **Какие действия мы выполняем для перехода из реального режима в защищённый?**

- Вычисляем базы для используемых дескрипторов структур.

- Вычисляем линейные адреса ТДС, ТДП.

- Устанавливаем смещение обработчиков прерываний клавиатуры в ТДП.

- Перепрограммируем контроллер.

- запрещаем все остальные прерывания.

- Открываем линию А20

- Меняем бит в регистре CR0.

(не совсем правильно - что-то ещё мы выполняем до вычисления базы дескрипторов)

**\*\*\* Какие две привилегированные команды мы используем?**

- LGDT (для загрузки ТДС)

- LIDT (для загрузки ТДП)

**\*\*\* Рассказать структуру дескриптора, объяснить, откуда берется 4Гб виртуальной памяти (было в лекции про организацию памяти)**

**\*\*\* Какие сегменты объявили и зачем**

(перечисляем в лоб - типа 4гб сегмент данных, кода и прочей хрени). Объяснить, что внутрях.

Сказать, что во-первых, есть 92h/98h в зависимости от типа регистра, что первое число - это первые 16 бит размера сегмента.

Сам размер составляется из вот этих 16 бит - младших разрядов - и младших четырех битов atr2 (для сегмента это старшие четыре бита) вместе они составляют размер.

В самом первом сегменте это FFFFFh - 1 Мб, но так как у нас проставлен флаг дробности 1 (старший разряд атр2), то у нас память выделяется не байтами, а 4кб-ными кусками.

**\*\*\* Зачем нужен 16битный сегмент кода. Ответ достоин кэпа -**

"Чтобы было где начать программу и где ее закончить" или что-то в этом духе.

**\*\*\* Как считаем число доступной памяти через этот 4гиговый регистр. Спросила, зачем пропускаем 1Мб.**

Ответ должен быть в следующем духе: в 1Мб кроме всего прочего у нас лежит биос.

Для биоса установлены права доступа readonly, а потому, если мы попытаемся туда что-то записать, то у нас будет ошибка по правам доступа.