# Вопросы по защищенному режиму:

1. Какую программу вы написали?  
   Написали управляющую программу с 0 уровнем привилегий.
2. Что пришлось создать в этой программе?  
   Две системные таблицы.
3. Где явно установлен соответствующий уровень привилегий?  
   2 бита (DPL) в первом атрибуте дескриптора сегмента.
4. Какие сегменты описали в глобальной таблице и почему, для чего?
   * 16-разрядный сегмент кода для реального режима;
   * 16-разрядный сегмент данных, для хранения данных (строки для вывода, например);
   * 32-разрядный сегмент кода для защищенного режима;
   * 32-разрядный сегмент данных, чтобы посчитать доступную память;
   * 32-разрядный сегмент стека.
5. Охарактеризуйте дескриптор сегмента дополнительной памяти, который вы описали.  
   ?.
6. Почему ваша таблица дескрипторов прерываний имеет такую структуру?  
   ?.
7. Что вы написали для исключений?  
   IDT, обработки для исключений (заглушки) и обработчики для прерываний клавиатуры и системного таймера.
8. Как адресуются аппаратные прерывания в защищенном режиме? (самый "ударный" вопрос)  
   см. рисунок из семинара 2.
9. Когда вызывается ваш обработчик от клавиатуры / системного таймера?  
   Для клавиатуры: вызывается при нажатии или отжатии кнопки на клавиатуре.  
   Для таймера: 18.2 раз в секунду.
10. Какие действия необходимо выполнить для корректного возвращения компьютера в реальный режим?  
    По коду:
    * Заперт MI;
    * \*far jmp;
    * Обнулить флаг pe в CR0;
    * Обновить сегментные регистры;
    * Перенастроить базовый вектор контроллера прерываний;
    * Восстановить маски контроллеров прерываний;
    * Восстановить состояние IDTR реального режима;
    * Закрыть линию А20 (Не обязательно?);
    * Разрешить MI и NMI.
11. Что такое теневые регистры? Для чего они включены в процессор? Какую информацию содержат?  
    «Теневые регистры находятся в процессоре. Нужны чтоб исключить постоянные обращения к физической памяти. Тк мы обращаемся к физ. памяти на каждой команде, а то и несколько раз, особенно если адресация косвенная.» (отвечать, наверное, лучше по-другому)  
    Из семинара 3: Задача теневых регистров – хранить информацию о текущем сегменте в самом процессоре, чтобы не обращаться в памяти (затратное действие). (чтобы избежать обращение к таблице дескрипторов)
12. Линия А20. Что произойдет с памятью, если при переходе в защищенный режим забудем открыть линию А20?  
    В реальном режиме линия A20 закрыта. Когда компьютер переводится из реального режима в защищённый, она должна быть принудительно обнулена. Иначе адреса с 1 в 20ой линии будут недоступны.
13. Что произойдет, если при возвращении в реальный режим забудем закрыть линию А20?  
    Если мы в реальном режиме откроем линию A20, нам станет доступно еще 64 Кб памяти (HMA)

**ТРЕБУЕТСЯ ПОЛНОЕ ПОНИМАНИЕ**

*/\**

*Привилегии != приоритеты  
Привилегии - 4 кольца защиты  
Приоритеты - приоритет процессов, потоков, который назначается и может быть пересчитан.   
Процессы выстраиваются в очередь в соответствии с приоритетом.*

*\*/*

# Семинар 3

Наша программа в защищённом режиме имеет нулевой уровень привилегий. В реальном режиме нету уровней привилегий.

Нет никакой защиты в реальном режиме (можем поменять вектор прерывания).

Привилегированные команды могу вызваться на определённом уровне привилегий. (То, что мы можем вызвать их в реальном режиме еще раз доказывает, что нету никакой защиты – нету уровней привилегий)

Таблица векторов прерываний – таблица реального режима. IDT – таблица защищённого режима.

Если первые 32 дескриптора (gate’а) отведены под исключения. С 32 по 255 определяются пользователем (User defined)

Нужно реализовать аппаратные прерывания (от таймера и клавиатуры).

Вектор прерывания = базовый вектор + номер IRQ. => нужно перепрограммировать базовый вектор ведущего контроллера на новый = 32. При этом мы больше никаких прерываний не обрабатываем. => на ведущий контроллер приходит только 2 сигнала (импульса): от клавиатуры и таймера. Соответственно больше никаких прерываний. На ведомый вообще никаких прерываний не приходит. Надо замаскировать все прерывания на ведомом контроллере (правила, следующие: 1-заперт, 0 - разрешение) => На ведущем FCh, на ведомом FFh. … обращается через порты (значит по адресу) значит через команды in и out. Порт 21h - ведущего, A1h - ведомого.

Написать сохранение масок. (размер маски байт):

mask\_master db 0

mask\_slave db 0

; Сохранение масок (Чтобы смогли их восстановить)

**in** **al,** 21h

**mov** mask\_master**,** **al** ; Ведущий.

**in** **al,** 0A1h

**mov** mask\_slave**,** **al** ; Ведомый.

; Перепрограммирование ведущего контроллера

; Т.к. (в з-р) первые 32 вектора зарезервированны для ообработки

; Исключений, аппаратным прерываниям нужно назначить другие векторы

**mov** **al,** 11h

**out** 20h**,** **al**

**mov** **al,** 32 ; это новый базовый вектор (был до этого 8)

**out** 21h**,** **al**

**mov** **al,** 4

**out** 21h**,** **al**

**mov** **al,** 1

**out** 21h**,** **al**

; Маска для ведущего контроллера

**mov** **al,** 0FCh ; 1111 1100 - разрешаем только IRQ0 И IRQ1 (Interruption Request - Запрос прерывания)

**out** 21h**,** **al**

; Маска для ведомого контроллера (запрещаем прерывания)

**mov** **al,** 0FFh ; 1111 1111 - запрещаем все!

**out** 0A1h**,** **al**

Восстановление масок:

; восстанавливаем маски контроллеров прерываний

**mov** **al,** mask\_master

**out** 21h**,** **al**

**mov** **al,** mask\_slave

**out** 0A1h**,** **al**

Важно запретить все прерывание: маскируемые и немаскируемые.

Маскируемые:

cli; Запрет аппаратных прерываний. (Маскируемых)

Немескируемые:

; Запрет немаскируемых прерываний. NMI

**mov** **al,** 80h

**out** 70h**,** **al**

Разрешение:

**sti** ; Резрешаем (аппаратные) прерывания

**xor** **al,al**

**out** 70h**,al**

Посылаются несколько команд, которые называется слово команды исполнения.

; Перепрограммирование ведущего контроллера

**mov** **al,** 11h ; СКИ1

**out** 20h**,** **al**

**mov** **al,** base\_vec ; СКИ2

**out** 21h**,** **al**

**mov** **al,** 4 ; СКИ3-IRQ2

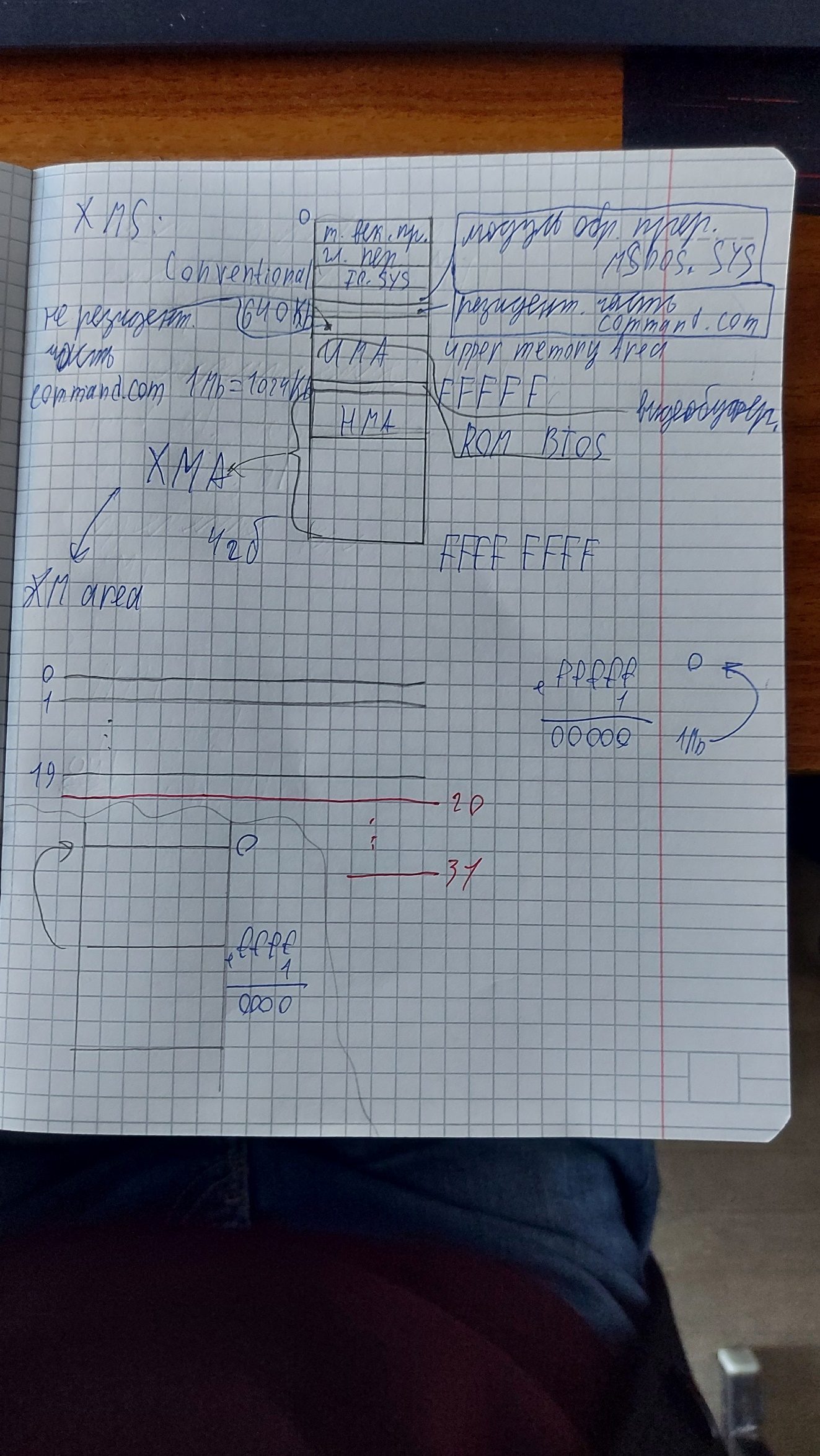
**out** 21h**,** **al**

**mov** **al,** 1 ; СКИ4 требует EOI

**out** 21h**,** **al**

### Линия A20

Спецификации XMS и EMS (XMS - extended memory specification, EMS - expanded memory specification)  
XMS – дополнительная память   
EMS – расширенная память (фактически – страничное преобразование)

В реальном режиме было 20 адресных линий. Было 2 типа заворачивания (огибания).

(Аппаратно поддерживает обратную совместимость)

В реальном режиме линия A20 закрыта. Когда компьютер переводится из реального режима в защищённый, она должна быть принудительно обнулена. Иначе адреса с 1 в 20ой линии будут недоступны.

Если мы в реальном режиме откроем линию A20, нам станет доступно еще 64 Кб памяти (HMA)

Задача теневых регистров – хранить информацию о текущем сегменте в самом процессоре, чтобы не обращаться в памяти (затратное действие). (чтобы избежать обращение к таблице дескрипторов)

Существует 2 кода (линия А20):

Открыть:

**mov** **al,**0D1h ; команда управления

**out** 64h**,al**

**mov** **al,**0DFh ; код открытия

**out** 60h**,al**

; открытие линии A20 (Быстрый)

**in** **al,** 92h

**or** **al,** 2

**out** 92h**,** **al**

В большинстве новых компьютеров, начиная с … (PS/2) имеется быстрый вариант открытия линии А20. Быстрый вариант исключает опрос. Быстрый вариант, считается не вполне надёжным. Поддерживается не всем платформами. Убедится заранее поддерживается или нет – невозможно => пользоваться надо 1ым вариантом.

Закрыть линию А20:

; закрытие линии A20

**mov** **al,**0D1h ; команда управления

**out** 04h**,al**

**mov** **al,**0DDh ; код закрытия

**out** 60h**,al**