# Семинар 2

ОС – это программа, GDT используется для адресации программы.

В GDT описывается … (?для физического адреса?)

**!\*!GDT содержит дескрипторы сегментов физической памяти.!\*!**

То-есть глобальная таблица дескрипторов, позволяет ?адресовать память?

Команды и данные которые находятся в оперативной памяти, находятся (доступ к ним) по адресу. Но такой адрес надо получить

Две схемы управления памятью:

1. Сегментами по запросу
2. Страницами по запросу

Предполагает выполнение преобразования памяти.

Сегментированный адрес…

Сегментный регистр смещается на 4 разряда и добавляется еще 4 байта.

В З-Р у нас есть начальный адрес начала сегмента (в дескрипторе из глобальной таблицы дескрипторов), а смещение мы берём из команды.

Обычно говорят: любая программа считает, что она начинается с нулевого адреса. Соответственно, отсюда получается, что мы получаем смещение из команды. Но мы должны понимать что это счётчик команд, или индексные регистры/указатели, или непосредственно …

Это смещение добавляется к … и получается линейный физический адрес по которому совершается обращение к памяти.

Процессоры Intel, также поддерживают локальные таблицы дескрипторов (LDTR – 16 разрядов => этот регистр, не может содержать полный адрес. В LDTR может лежать селектор) Локальная таблица описывает виртуальное пространство, локального процесса. Будет столько локальных таблиц, сколько процессов.

Во 2ой лабе:

* Работу с таймером и клавиатурой
* Определение

Наши JDT и IDTR – системные таблицы

Убрать сброс процессора.

В этой программе

Описаны 16-разрядные сегменты, а защищённый режим 32-разрядный, соответственно

**Задание**: расписать дескрипторы сегментов на двоичные коды и какие это сегменты.

Мы анализируем код из Рудакова-Финагенова.

В этой программе для всех 4ёх регистров установлена граница (lim) = FFFFh = 2^16=64 Кб. Потому что регистре в реальном режиме 16-разрядные.

2^20 = FFFFFh – 1 Мб –

Р-Ф ?Забыли про еще 4-е разряда в атрибутах?

Теневые регистры

С каждым сегментным регистром в процессоре сопоставлен теневой регистр. В котором при обращении к сегментному регистру, записывается информацию из …. Сделано это для того, чтобы исключить обращение к таблице JDT (которая находится в оперативной памяти). Дело в том, что О.П. отстаёт по производительности на порядок, обращение к ОП – трудозатратное действие (сущ. Понятие цикл обращение к памяти – требует определённое кол-ва тактов.

(Как часто обращается к ОП – на каждой команде а то и больше. + преобразование адреса)

Чтобы этого избежать, информация с дескриптора, записывается в теневой регистр. Теневые регистры находятся, непосредственно в процессоре (нету обращение к ОП – исключает обращение к таблице JDT)

Программа в Р-Ф кривая! Адресация — это важно!

Поскольку в реальном режиме, смещение не может превышать FFFF устанавливается граница, но в этом нет смысла, т.к. описанные сегменты и так не выходят за эту границу. (закомментировать эти 4 строки)

В нашей программе - Определить объём доступного адресного пространства.

Что для этого нужно сделать? (в З-Р можем адресовать 4 Гб) Для того чтобы адресовать нужно объявить дескриптор. 1ый Мб пропускается, со 2ого Мб-та … записываем туда сигнатуру, сравниваем со своей сигнатурой. Если сигнатуры совпали, то это память. И естественно инкрементировать счётчик. Нужно объявить дескриптор сегмента памяти 4Гб (бит гранулярности = 1, для чтения-записи)

В этих дескрипторах должны быть

* 16-разрядный сегмент кода
* 16-разрядный сегмент данных
* 32-разрядный сегмент кода
* 32-разрядный сегмент данных (для определение доступного объёма памяти)
* 32-разрядный сегмент стека (когда возвращаемся в реальный режим, нужно вернуться к стеку для реального режима)

Мы должны написать 2 обработчика:

* Для таймера
* Для клавиатуры

## Прерывания в защищённом режиме

Есть IDTR (32-разрядный) – в этом регистре находится начальный адрес таблицы дескрипторов прерываний.

Все дескрипторы по 8 байт.

Первые 32 дескриптора IDT отведены под исключения.

Определены следующие исключения (некоторые):

* Нулевое - 0 – Деление на ноль (div error)
* Восьмое - 8 – Double Fault (Происходит если выполняется исключение и/или маскируемое прерывание и происходит ошибка – паника, компьютер выключается)
* 13-е – General Protection (Общая защита – должно обрабатываться специальным образом. В нашей программе на все исключения мы пишем заглушки. На 13-е пишем специальную заглушку)
* 11-е – Segment not present (Возникает когда сегмент отсутствует. Нашей программы это не касается, это касается управления памятью)
* 14-е – Page Fault (Страничное прерывание. Возникает, когда процессор обращается к странице или данным, которые отсутствуют. Обрабатывая это исключение, нужно загрузить …)

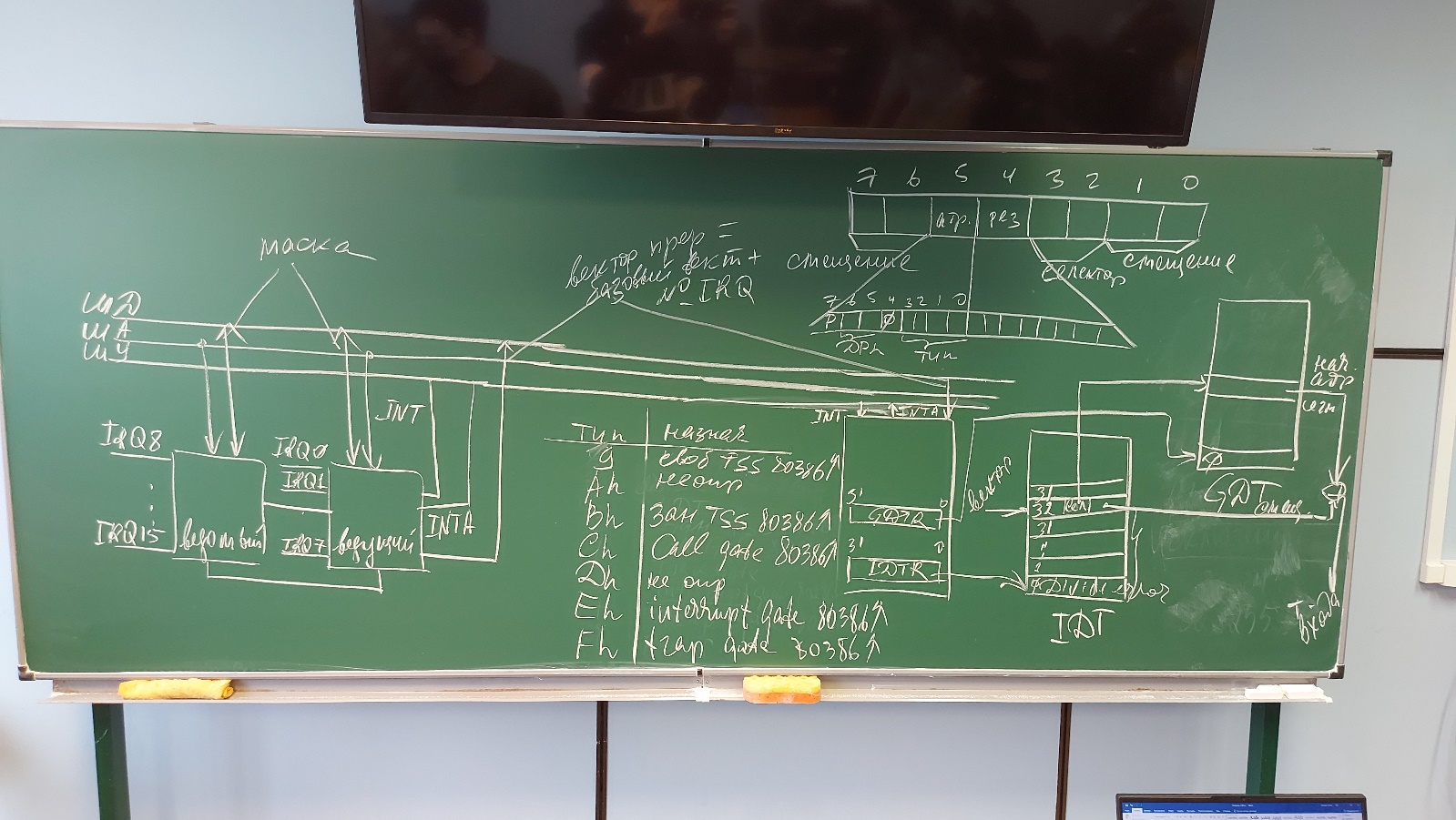
Всего определенно 19 исключений.

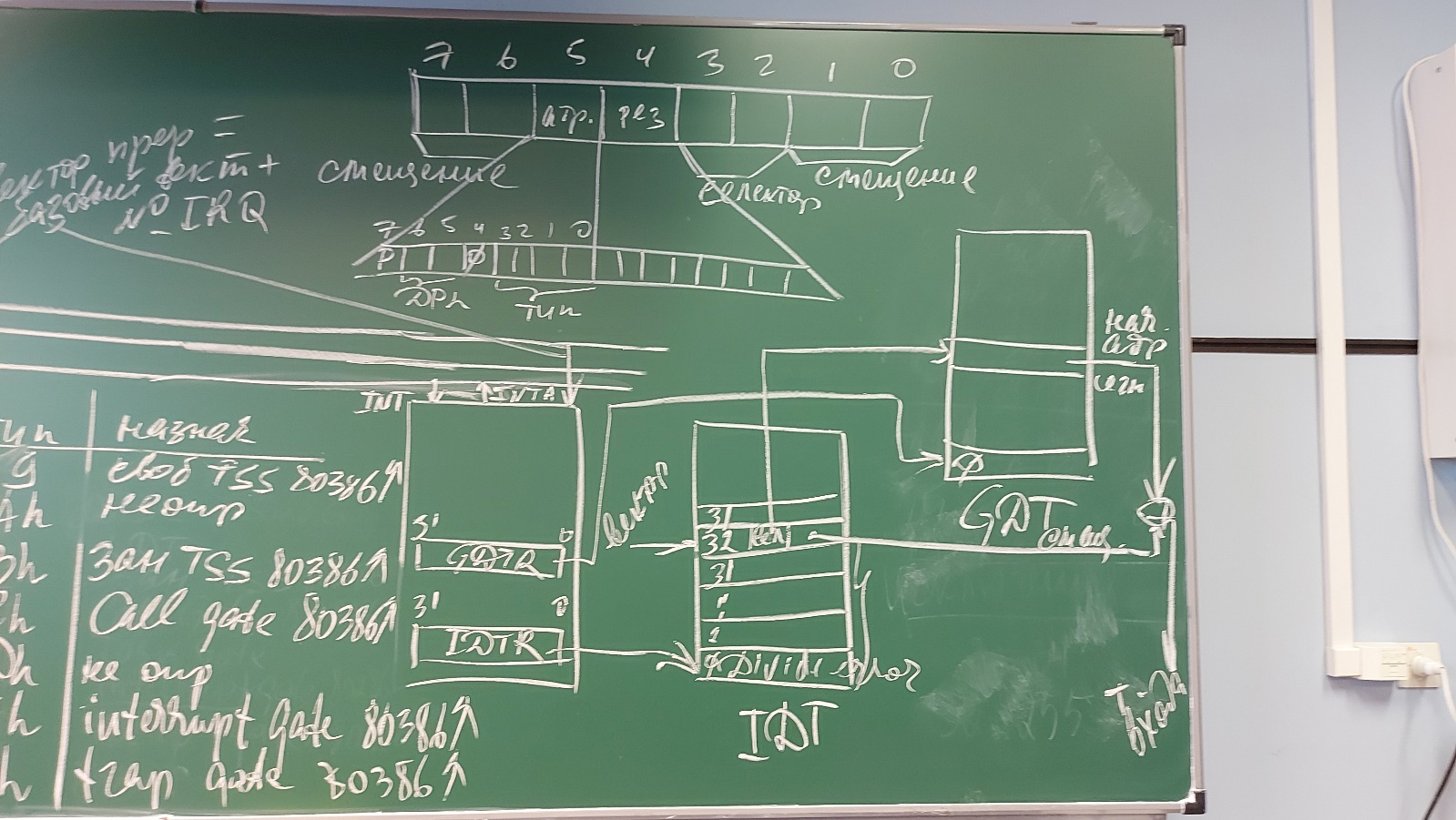
Исключения с 20-ого по 31-е зарезервированы.

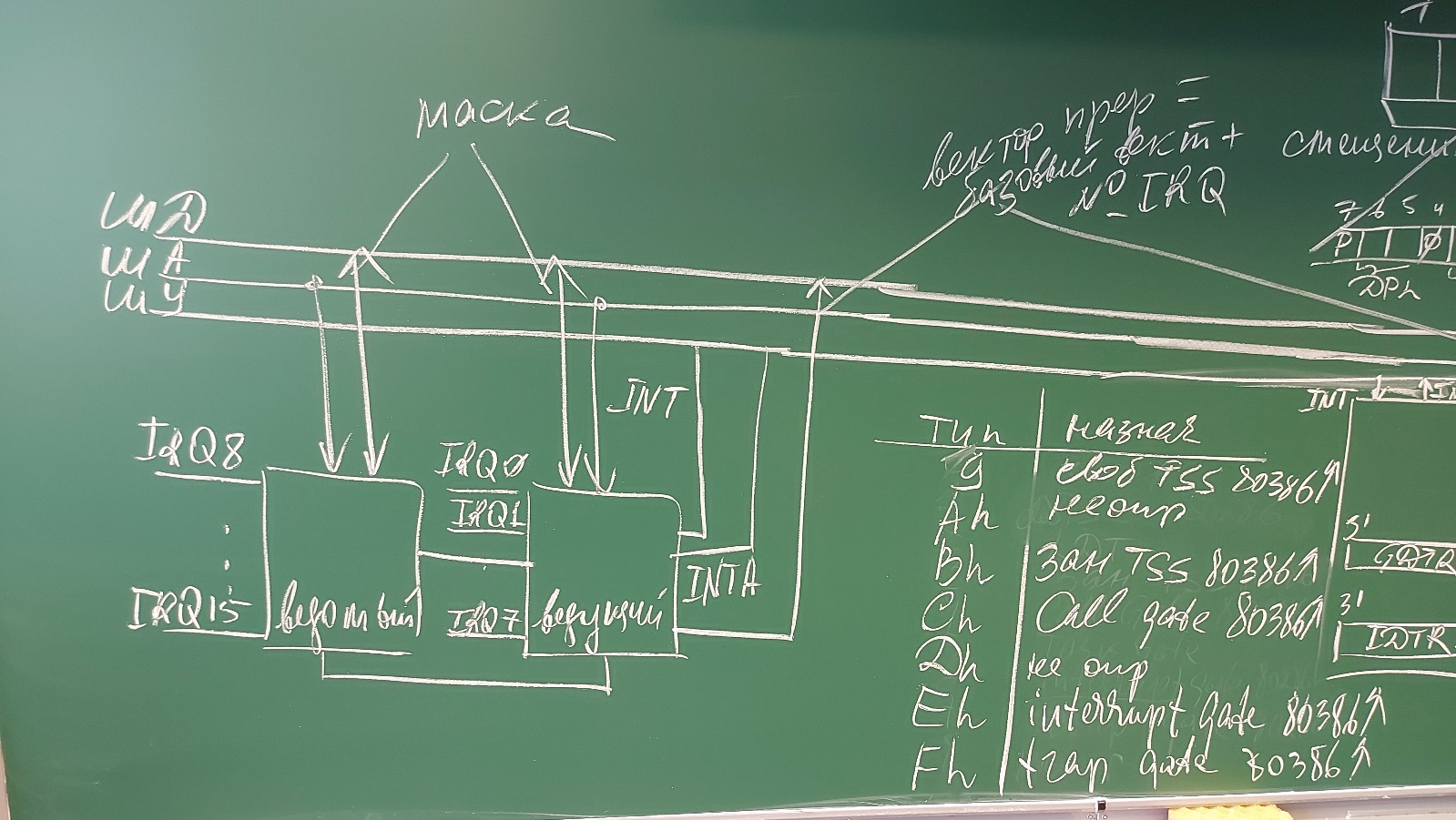
И с 32 по 255 определяются пользователем (User defined)

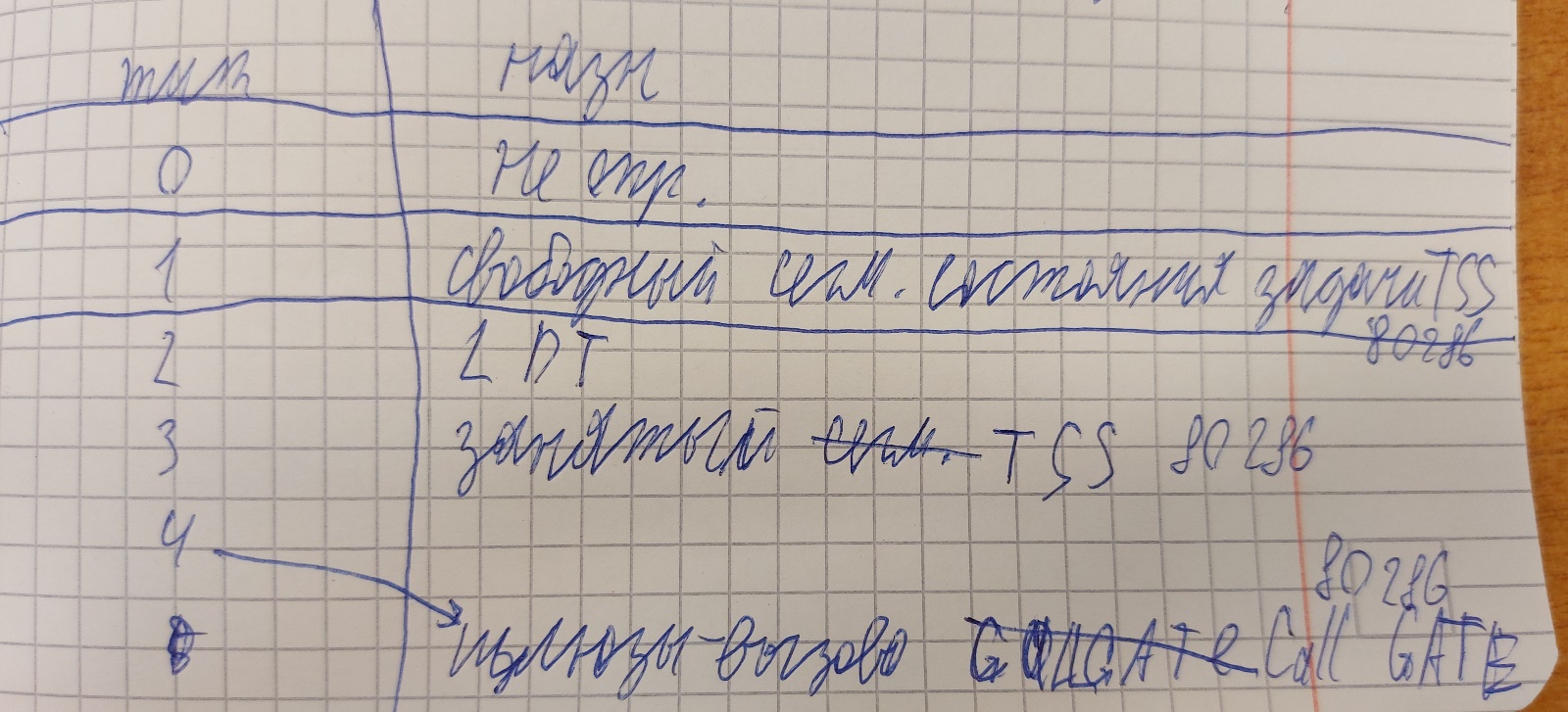
Также начинается с нулевого дескриптора (Он не пустой - деление на ноль).

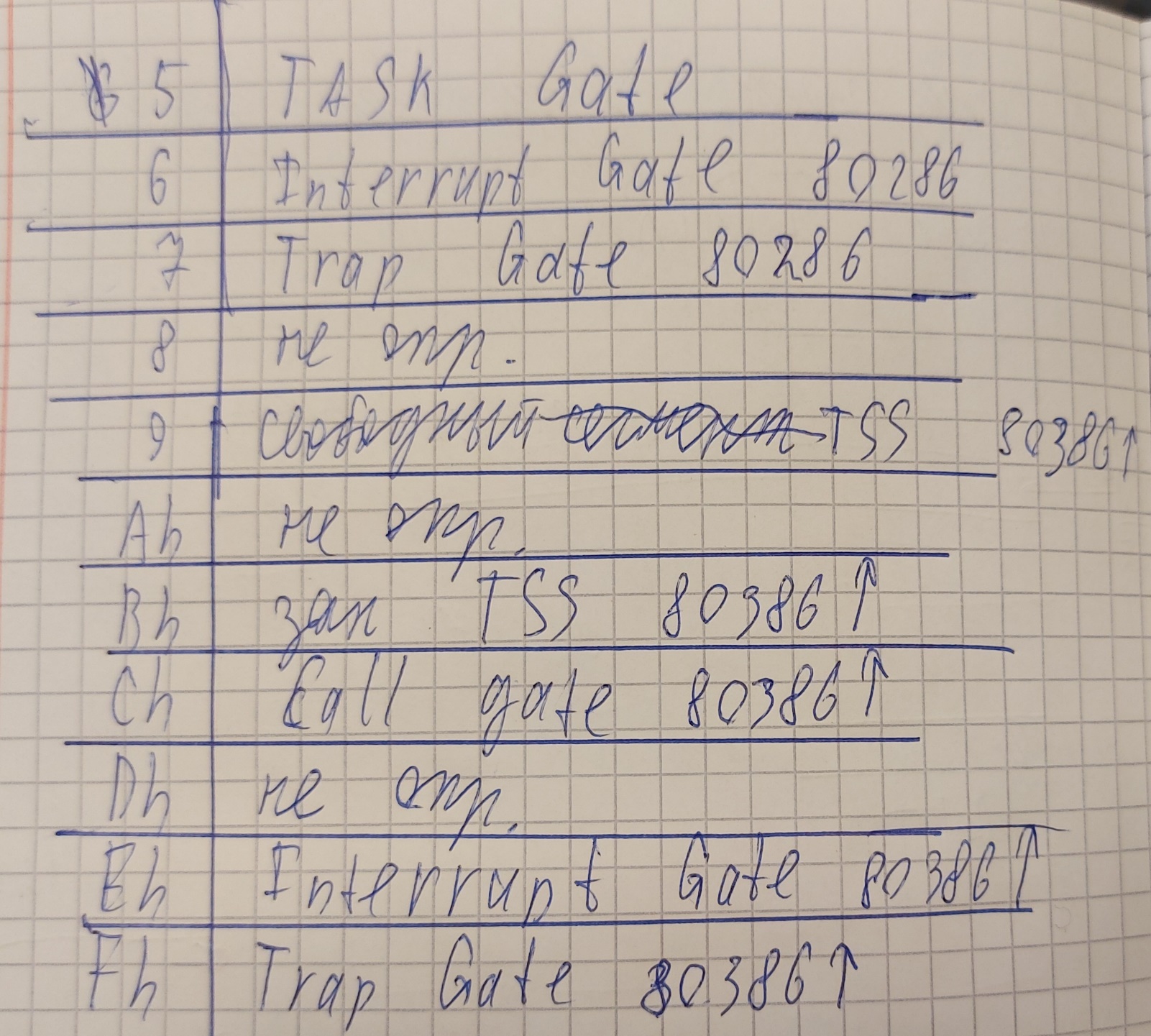
Номер исключение умножается на 8.

Нам нужно написать обработчики для прерываний таймера и клавиатуры (это аппаратные прерывания). Самое простое – через программируемый контроллер прерываний.  






Нам нужно перепрограммировать контроллер прерываний.  




Таким образом обработка аппаратных и программных прерываний.

***Любимый вопрос: почему мы пропускаем 1ый Мб (когда определяем …)?***