БИЛЕТ № 30
1) Назначения cash.

Под словом **кэш** (cash – "наличные"), этимологию которого мы не будем здесь рассматривать, обычно понимают область быстрой памяти, содержащую копию данных, расположенных где-либо в более медленной памяти, предназначенную для ускорения работы вычислительной системы. Мы с вами сталкивались с этим понятием при рассмотрении иерархии памяти. В базовой подсистеме ввода-вывода не следует смешивать два понятия, буферизацию и кэширование, хотя зачастую для выполнения этих функций отводится одна и та же область памяти. Буфер часто содержит единственный набор данных, существующий в системе, в то время как кэш по определению содержит копию данных, существующих где-нибудь еще. Например, буфер, используемый базовой подсистемой для копирования данных из пользовательского пространства процесса при выводе на диск, может в свою очередь применяться как кэш для этих данных, если операции модификации и повторного чтения данного блока выполняются достаточно часто.

Функции буферизации и кэширования не обязательно должны быть локализованы в базовой подсистеме ввода-вывода. Они могут быть частично реализованы в драйверах и даже в контроллерах устройств, скрытно по отношению к базовой подсистеме.

Кэширование. Для минимизации количества обращений к диску применяют блочный или буферный кэш. Кэш – набор блоков, логически принадлежащих диску, но хранящиеся в оперативке. Происходит захват всех запросов чтения к диску и проверке требуемой информации в КЭШе. Если блок присутствует – запрос чтения происходит без обращения к диску. Иначе блок считывается с диска в кэш, оттуда копируется по нужному адресу памяти

Кэш процессора обычно является частью аппаратуры, поэтому менеджер памяти ОС занимается распределением информации главным образом в *основной* и внешней памяти компьютера. В некоторых схемах потоки между *оперативной* и внешней памятью регулируются программистом (см. например, далее *оверлейные структуры*), однако это связано с затратами времени программиста, так что подобную деятельность стараются возложить на ОС.

Кэш-память - это способ организации совместного функционирования двух типов запоминающих устройств, отличающихся временем доступа и стоимостью хранения данных, который позволяет уменьшить среднее время доступа к данным за счет динамического копирования в "быстрое" ЗУ наиболее часто используемой информации из "медленного" ЗУ.

Доп инфа. Каким образом система отслеживает попадание в КЕШ

Кэш-память SRAM используется для хранения данных, которые непосредственно требуются процессору. Статистический анализ работы современных компьютеров показывает, что около 90% всех данных, которые запрашивает процессор, обычно находится в кэш-памяти; эта величина называется коэффициентом кэш-попадания. Если процессор находит нужную ему информацию в кэш-памяти, это называется кэш-попаданием, иначе ему приходится искать нужную информацию в DRAM; такая ситуация называется кэш-промахом.

Поскольку кэш-память SRAM работает быстрее, чем память DRAM, коэффициент кэш-попадания пропорционален эффективному времени ожидания всей системы памяти. Если обозначить коэффициент кэш-попаданий через H, а время ожидания кэш-памяти и DRAM - через Tc и Tm соответственно, то эффективное (среднее) время ожидания системы памяти, включающей в себя кэш-память, определяется как : $Teff= H * Tc + (1-H)*Tm$

Увеличение скорости, связанное с наличием кэш-памяти, определяется как отношение Tm к Teff и экспоненциально увеличивается с возрастанием H: $S=Tm/Teff = 1/(1-H(1-Tc/Tm))$

P.S.: надеюсь у тебя хватит ума не писать формулы??
2) Блокировка записи.
Смысл блокирования и разблокирования записей.

При открытии файла выделяется буфер, равный размеру кластера.

При загрузке ОС создается таблица дескрипторов файлов(какому процессу принадлежит файл, буфер файла). Каждому буферу подчинен указатель на текущую позицию в файле.

Разблокирование – считывание кластера целиком, из него выбирается логическая запись.

Блокирование – при любой записи в файл запись производится в буфер до тех пор, пока он не будет заполнен, тогда весь кластер пишется на диск. Когда закрываем файл, то независимо от заполненности буфера, он скидывается в файл.

ВОПРОСЫ RANDOM
1)Особенности реализации переключения потоков в многопрограммных ОС.
2) Схема формирования эффективного адреса в процессе runtime.
3) RAID-массивы. Сравнение.
4) Методы эффективного управления внешними единицами информации.
5) Что такое когерентность.
Когерентность

Для программиста различают явный и неявный доступ к данным. Тогда:

1) Явное размещение данных – явнй доступ (все операции send/receive заданы явно).
2) Неявное размещение – неявный доступ (доступ к данным прозрачный для программиста).
3) Неявное размещение страниц в памяти – явное указание доступа (используется разделяемое множество страниц на разных устройствах).
4) Явное размещение – неявное указание доступа, используется команда Load-store.

Реализация когерентности в однопроцессорных машинах:

Три способа организации памяти:

- с прямым отображением;
- частично ассоциативная;
- ассоциативная.

Для обеспечения когерентности необходимо выбрать правила согласования:

1. Со сквозной записью (самая быстрая).
2. С обратной записью (при кэш промахе или при смене страниц).
3. С буферизацией (происходит накапливание в буфере).

Сквозная – изменения переносятся вниз, как только они возникают в КЭШах

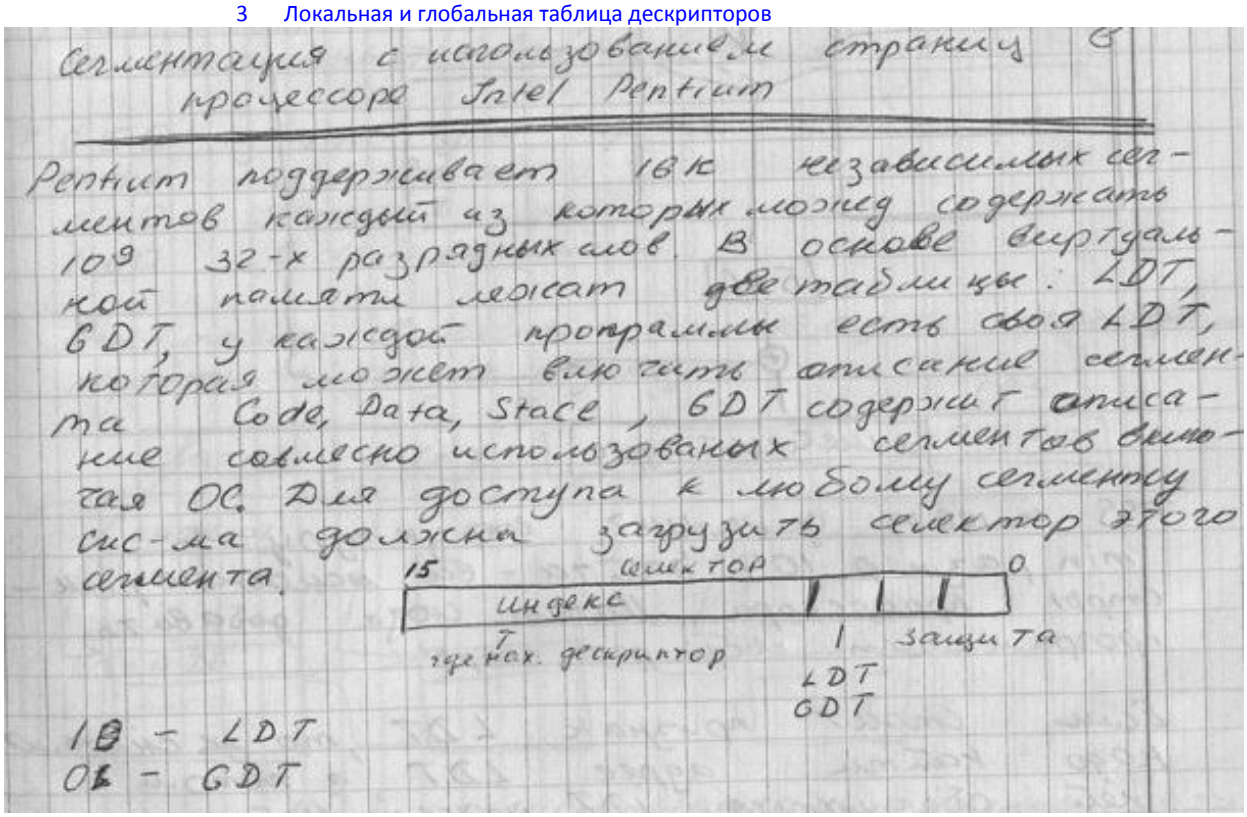
6) Переход процессов из одного состояния в другое.

7) Формирование эффективного адреса в safe mode

8) Может ли Windows работать в реальном времени.

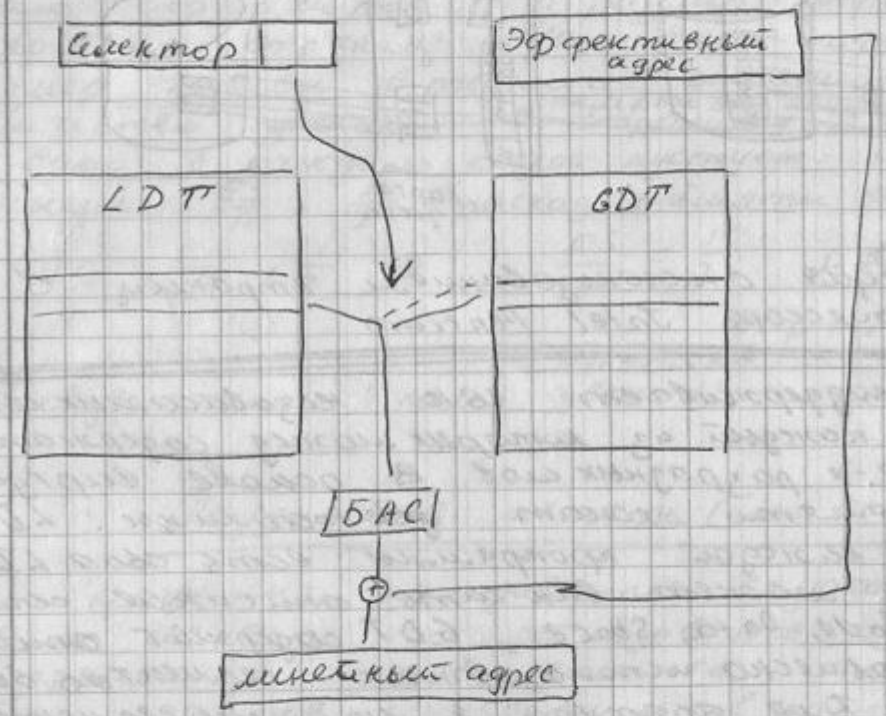
9) Уровень сигналов и программа обработки прерывания.

3 Локальная и глобальная таблица дескрипторов



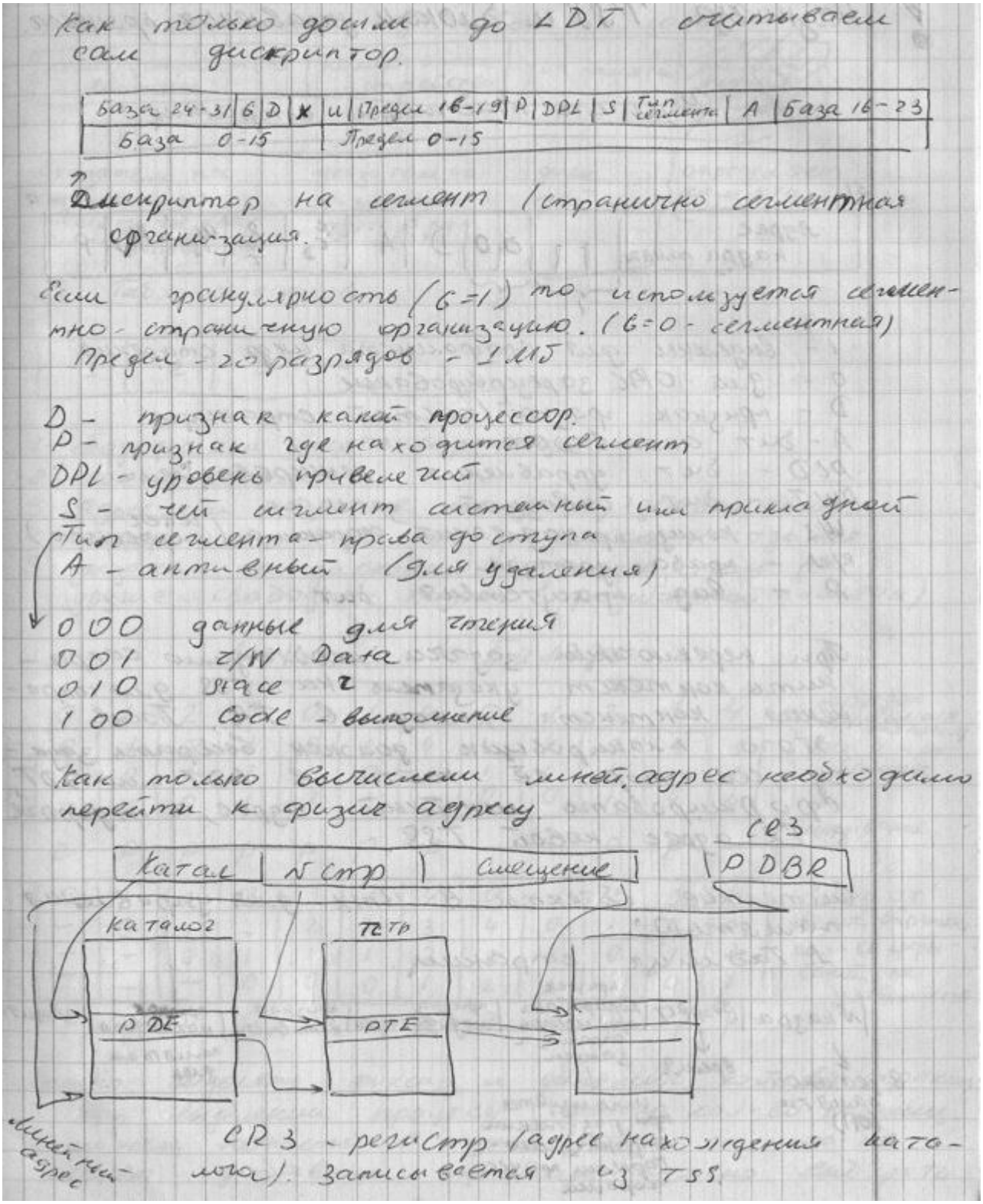
Для поиска GDT и LDT используется регистры GDTR, LDTR, IDTR, TR - содержит указатели на TSS текущей задачи. В TSS сохраняется контекст задачи при смене задачи.

Дискриптор находится в GDT или LDT. В соответствии с селектором, а именно индексом требуется вычислить линейный адрес сегмента, который использ. для вычисления физич адреса.



TSS имеет динамич структуру и min размер 104 байта - все основные регистры процессора. Может сюда добавить программист свои регистры!

Если стоит признак LDT, то сначала надо найти адрес LDT а потом к ней обращаться. LDT можем найти только через GDT.



4) Назначение системы ввода-вывода
Ответ:
Доп.инфа:

Существенная часть операционной системы занимается вводом-выводом. Операция ввода-вывода может выполняться тремя способами. Во-первых, при помощи программного ввода-вывода, при котором центральный процессор вводит или выводит каждый байт или слово, находясь в цикле ожидания готовности устройства ввода-вывода. Второй способ представляет собой управляемый прерываниями ввод-вывод, при котором центральный процессор начинает передачу ввода-вывода для символа или слова, после чего переключается па другой процесс, пока прерывание от устройства не сообщит ему об окончании операции ввода-вывода. Третий способ заключается в использовании прямого доступа к памяти (DMA), при котором отдельная микросхема управляет переносом целого блока данных, и инициирует прерывание только после окончании операции переноса блока.

Ввод-вывод можно разбить на четыре уровня иерархии: процедуры обработки прерываний, драйверы устройств, независимое от устройств программное обеспечение ввода-вывода и библиотеки ввода-вывода и спулеры, работающие в пространстве пользователя.

Драйверы устройств управляют деталями работы устройств и предоставляют однородные интерфейсы к остальной части операционной системы. Независимое от устройств программное обеспечение ввода-вывода занимается буферизацией и сообщением об ошибках.

---

5) *Правила формирования рабочего множества*

Конспект: В виду того,, что проц. Работает только с кешами, а не с ОП и для того, чтоб в кеше находилась только полезная часть проги, ввели понятие „рабочее множество”. В него должно быть включено наиболее активное использование страницы. Т.е. сис-ма должна автоматич. фиксировать «РМ».

Когда у сис-мы уменьшается частота стр. прерываний, сис-ма считает, что РМ сформировано.

Теория РМ основана на 2-х принципах:

1) временной локальности – если прога загружается, то она будет выполняться хоть какой-то $\Delta\tau$
2) пространственной локальности – если загрузили какой-то участок проги и выполнили некий участок проги, то след. команду будем выбирать близлежащую.