1 Иерархическая структура памяти

1.1 Вспомогательная память

Не важно, насколько огромно количество основной памяти: ее все равно не хватает людям. Строго говоря, человек будет удовлетворен в этом плане только тогда, когда ограничений на память не будет существовать. С развитием технологий людям приходит в голову сохранить такие данные, которые раньше не представлялось возможным.

Иерархическая структура памяти является традиционным решением проблемы хранения большого количества данных, она приведена на рисунке.

Рис. 2.15. Пятиуровневая организация памяти

По мере продвижения по структуре сверху вниз возрастают три параметра. Во-первых, увеличивается время доступа.

Во-вторых, увеличивается объем памяти.

В-третьих, увеличивается количество битов, которое вы получаете за 1 доллар.

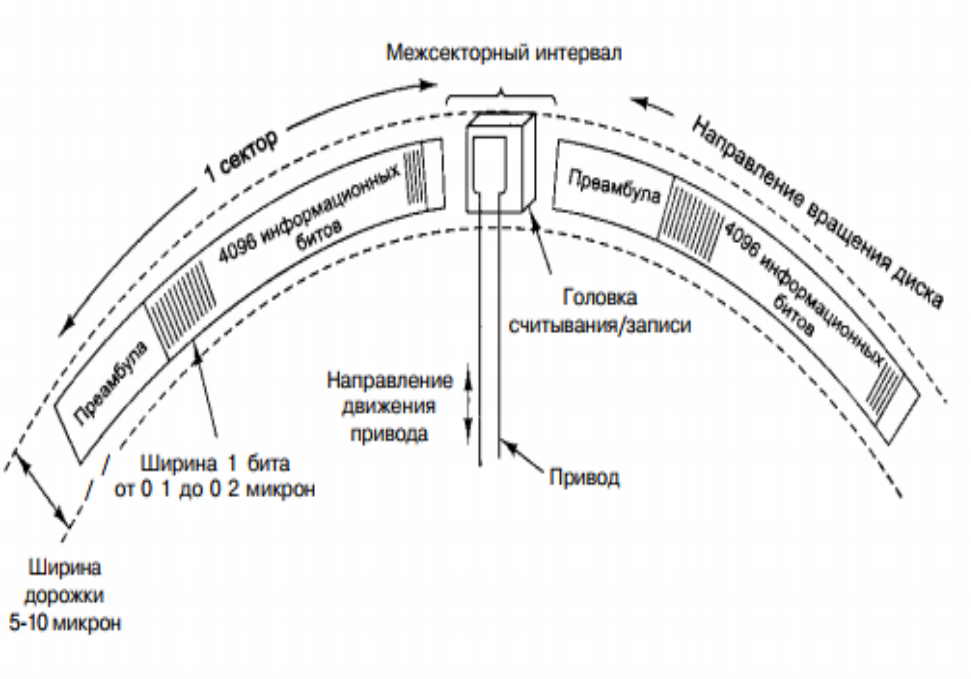
2 Виды хранителей информации

2.1 Магнитные диски

Магнитный диск состоит из одного или нескольких алюминиевых дисков с магнитным слоем. Сейчас их диаметр составляет от 3 до 12 см, причем этот параметр продолжает уменьшаться. Головка диска, содержащая индукционную катушку, двигается над поверхностью диска, опираясь на воздушную подушку.

Когда головка проходит над намагниченной областью, в ней (в головке) возникает положительный или отрицательный ток, что дает возможность считывать записанные ранее биты. Поскольку диск вращается под головкой, поток битов может записываться, а потом считываться.

Дорожкой *называется круговая последовательность битов, записанных на диск за его полный оборот.* Каждая дорожка делится на сектора фиксированной длины. Каждый сектор обычно содержит 512 байтов данных(4кб сейчас, ну про это нужно на лекции послушать). Перед данными располагается **преамбула** (preamble), которая позволяет головке синхронизироваться перед чтением или записью. После данных идет код с исправлением ошибок (код Хэмминга или чаще код Рида—Соломона, который может исправлять много ошибок, а не только одиночные). Между соседними секторами находится межсекторный интервал. Ширина дорожки с каждым годом уменьшается, кажется сейчас на 1 см приходится 10к дорожек. У всех дисков есть кронштейны, они могут перемещаться туда и обратно по радиусу на разные расстояния от шпинделя, вокруг которого вращается диск.



Плотность записи битов на концентрических дорожках различная, в зависимости от расстояния от центра диска. Такие диски называются **"Винчестер"** возможно потому, что когда они впервые были выпущены IBM, они имели 30 Мбайт фиксированной памяти и 30 Мбайт сменной памяти, а у пиндосов было популярное ружье Винчестер 30-30.

Большинство дисков состоят из нескольких пластин, которые друг над другом находятся. Множество дорожек равноудаленных от центра называют **цилиндром**. (Что за тупость) На каждой пластине своя независимая головка.

Чтобы считать что-то, головка должна для начала это найти! Сначала привод отпраляет ее на нужную дорожку, а потом она ждет некоторые время, пока под ней не окажется нужный сектор. Это процесс называется, неожиданно, **поиском**.

Современные венчестеры вращают диски со скоростью >200 оборотов в секунду, поэтому скорость записи превышает 40 Мбайт в секунду. Строго говоря, с такой скоростью вращения они могут немного механически менять свои свойста, расширяться, например. Поэтому иногда происходит рекалибровка головки, чтобы она учитывала это расширение. Это может быть плохо, к примеру, при воспроизведении музыки, для которого требуется более или менее непрервный поток данных. Для таких

целей специально делают аудио-видео диски.

А теперь очевидный вопрос, если диск крутится с постоянной угловой скоростью, то как быть с тем, что ближе к центру за одно и то же время пройдет меньшее расстояние, чем ближе к краю? Раньше, для решения этой проблемы плотность записи по ближе к краю уменьшали, тем самым за одно и то же время количество информации на всех дорожках считывалось одинаковое, но такой подход оказался не самым эффективным. Сейчас используют другую стратегию: цилиндры делятся на сектора, чем ближе к центру, тем их меньше. Сектора имеют одинаковый размер. Такое представление усложняет запись информации, но позволяет куда эффективней использовать место.

С диском связан так называемый контроллер - микросхема, которая управляет диском. Некоторые контроллеры содержат целый процессор. В задачи контроллера входит получение от программного обеспечения таких команд, как READ, WRITE и FORMAT (то есть запись всех преамбул), управление перемещением рычага, обнаружение и исправление ошибок, преобразование 8-битных байтов, считываемых из памяти, в непрерывный поток битов и наоборот. Некоторые контроллеры производят буферизацию совокупности секторов и кэширование секторов для дальнейшего потенциального использования, а также устраняют поврежденные секторы. Это нужно, потому что некоторые сектора повреждаются, то есть остаются постоянно намагниченными. За этим следит контроллер.

2.2 Оптические компакт диски

Изначально эти штуки создавались для хранения аудио-видео информации, создавала их компания Philips совместно с голивудом. Их плюс был, что они небольшие, но хранят потенциально огромное количество информации, а так же их легче продавать, чем магнитные диски с фильмом(ну зачем, нам же не нужно переписывать).

Выглядит компакт диск, вы знаете как. Круг, на поверхности которого есть тонкая отражающая поверхность. Изначально изготавливался так: инфрокрасным лазером по спирали прожигали в некоторых местах отверстия, **впадины(pit)**, таким образом дорожка выглядила как последовательность впадин и **площадок(land)**. Когда же это все считывалось, впадины отражали свет иначе, нежели площадки. Сначала подумали, хм, давайте использовать впадину за 0, а площадку за 1, но это оказалось не очень точно, поэтому решили считать за 1 переход из **впа-**

дины в площадку или **площадки в впадину**, а отсутствие перехода за 0. Примерно выглядит это так:

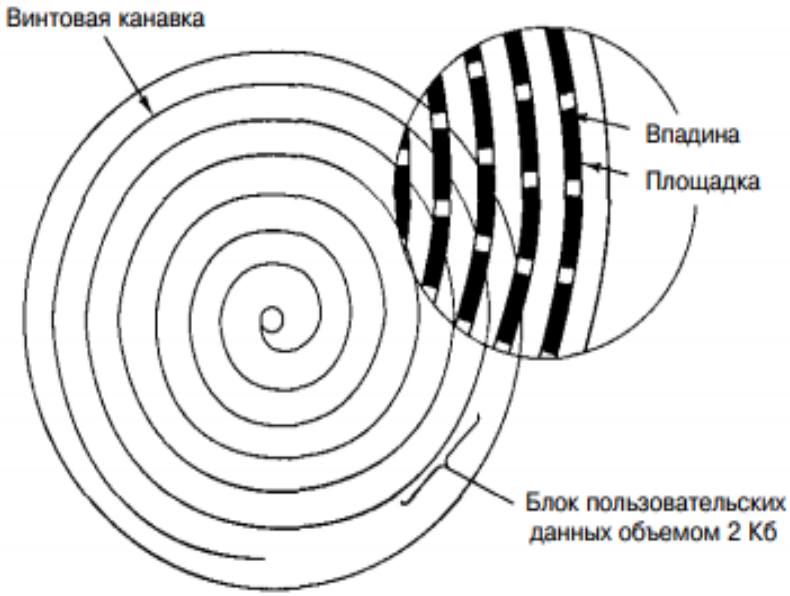


Рис. 2.19. Структура записи компакт-диска

Из-за того, что в оптическом диске, в отличие от магнтного, одна спиралевидная дорожка, то при равной угловой скорость головки, скорость чтения к концу диска выше, чем к началу.

Вот то, что я описал только что это аудио-оптические диски, а умные челы из Philips решили использовать для этого не только порнуху, но и адекватные данные, выпустили стандарт и назвали эт **CD-ROM (Compact Disk - Read Only Memory)**. Такая тема была востребованной, потому что стоимость такого компакт диска была 1 доллар за штуку. Единственная разница была в том, что в случае аудио-видео-порно-дисков нам не нужно было ловить ошибки в битах, потому что человеческое ухо не смогло бы распознать, если бы в вашем музоне 1 бит отличался бы от оригинала, а вот компьютерщики с этим запарились. Они кодировали **каждый байт 14-битным символом**, что позволяло использовать кодирование Хэминга для 8-битного байта и еще 2 в придачу. Вообще, говорят, что там используется какая-то более крутая система, которая 16 битами кодирует 8 бит, но я не нашел где про это прочитать :(

Далее, 42 подряд идущих таких символов образуют **фрейм** из $42 * 14 = 588$ бит. Такой фрейм содержит 192 бита(24 байта) информации, , а 396

битов отдается под исправление ошибок на уровне фреймов.

Потом, каждые 98 (видимо тестировали rand(), когда выбирали числа) фреймов объединяются в один **сектор**. Каждый сектор начинался с **преамбулы** в 16 байтов, первые 12 из которых - $O0FFFFFFFFFFFFFFFFFFFFO0$(в шестнадцатиричной системе), так проигрыватель понимал, что сектор начался. Следущие 3 байта содержали номер сектора, потому что данные записаны одной спиралевидной дорожкой и искать за линию не очень хотелось бы, а тут можно бинпоиск, к примеру, замутить (спойлер, челики не догадались), а последний типа диска. Програмное обеспечение высчитывает, куда примерно ткнуть, а потом ищет преамбулу. Так же, 288 байт отводилось на исправление ошибок на уровне секторов. Итого, сектор содержит $98*24-288-16 = 2048$ байт информации. Тип диска влиял на то, нужны ли на эти 288 байт на исправления ошибок на уровне секторов, или мы можем забить на это и использовать их на инфу. Так же, стоит отметить, что мы имеем 2048 байт инфы, а храним $98 * 588/8 = 7203$ байт. Значит эффективность всего $2048/7203 = 28$ процентов примерно. Не так уж и круто. Такие диски хранили 512 секторов, то есть они как бы имели $512 * 7203 = 3687936$ байт, но информации там было $512 * 2048 = 1048576$ байт.

Односкоростные считывающие устройства читают со скоростью 75 секторов в секунду. ПОСЧИТАТЬ СКОЛЬКО ЭТО БАЙТОВ В СЕКУДНУ ОСТАВЛЮ В КАЧЕСТВЕ УПРАЖНЕНИЯ.

Можно так же добавить, шобы выебнуться перед скаковым, что для поддержки совместимости на разных компуктерах челы договорились о файловой системе, которую назвали **High Sierra**(да да, как макос). Про нее можно почитать в гугле.

CD-ROM сначала записывались дорогостоящим оборудованием и на заводах, но прогресс не стоит на месте. Появились **CD - R(Compact Disk - Recordable)**. Это были болванки, на которых была размечена спиралевидная дорожка, чтобы устройства понимали куда писать. Работало это так: устройство, которое записывало информацию лазером могло менять отражающую способность поверхности, какая-то химия, короче. Грубо говоря, вся поверхность была отражающая, а поверх ее был краситель, который ее закрывал, лазер выжигал краситель и отражающая поверхность проявлялась. Такие диски можно было дописывать. Расскажу байку. Значица так. Так как раньше, в соответствии со стандартом на диске была специально тема, которая называлась **(Volume**

Table of Contents — оглавление диска). Но, если мы что-то дозапишем в диск, это херотень должна поменяться, а менять-то мы ничего не можем, можем только дописывать. Ну, так сказать, запилим костыль. Давайте когда мы что-то дописываем, мы будем копировать нашу VToC и дописывать в нее данные, которые дописали. А когда будем считывать, сначала найдем самую последнюю VToC, а потом относительно нее будем ориентироваться. Такая штука называется **многосессионностью**. Плюс в том, что мы можем имитировать удаление с диска. Для этого допишем что-нибудь, а предыдущую таблицу скопируем не полностью, или вообще не скопируем. Операционная система подумает, что диск соотвествует последней таблице и будет игнорировать ненужные нам файлы. Каждая такая новая дорожка со своей VToC должна записываться с постояной скоростью, а жесткие диски, от которых нам поступают данные работают не с постоянной скоростью. Таким образом, если просто начать записывать данные, прося их у диска, велик шанс того, что наш CD-R привратиться в подставку для пиваса. Ну гениальные челы на програмистах придумали такую вещь: давайте сначала скопируем все нужные нам данные на жесткий диск последовательно, чтобы все работало с одинаковой скоростью. Такое плохо по 3 причинам

1. Удваивает время записи
2. Требует 650 мбайт на жестком диске
3. Не защищает от описанной ранее рекалибровки.

Короче программисты дебики.

CD-RW(Compact Disk - Re-writable - суть та же, но поверхность другая. Теперь она состоит из каких-то химических сплавов, что позволяет перезаписывать данные. У головки устройства есть 3 режима.

1. Высокая мощность, которая расплавляет сплав переводя его из кристалического состояния, отражающего, в аморфное, слабоотражающее.
2. Средняя мощность, расплавляет сплав, но не так сильно(:)))))), переводя его из аморфного состояния в кристаллическое.
3. Слабая мощность, ничего не делает со сплавом, но позволяет считывать его состояние.

DVD (Digital Video Disk раньше, а сейчас Digital Versatile (универсальный) Disk Ну, это то же самое, только с повышенной емкостью, которая достигается следущими штуками:

1. Впадины меньшего размера (0,4 микрона вместо 0,8 микрона, как у обычного компакт-диска).
2. Более плотная спираль (0,74 микрона между дорожками вместо 1,6 микрона).
3. Красный лазер (с длиной волны 0,65 микрона вместо 0,78 микрона).

Все это дало 7-кратное увеличение емкости.(4.7 Гбайт). Всю эту тему придумали совместно с Голивудом, которым 4.7Гбайта все еще было мало.(4,7 Гбайт может вместить полноэкранную видеозапись на 133 минуты с высокой разрешающей способностью (720x480) вместе с озвучиванием на 8 языках и субтитрами на 32 других языках.). Алчные микрочелы решили сделать **двуслойную** поверхность, которая позволяла в зависимости от типа лазера записывать на одну и ту же поверхность разные данные, а еще, решили писать на обеих сторонах диска, что увеличило еще в два раза. Итого, с погрешностью на реализацию двойной поверхность такое создание смогло уместить в себе аж 17 Гбайт данных.

Blue Ray Disk Ну короче это не интересно, то же самое, что и DVD, только лазер другой, тоньше.

2.3 RAID массивы

RAID - Redundant Array of Inexpensive Disks(Избыточный массив недорогих дисокв)

У, бля... Ну короче, это тема для бомжей. Поцаны решили, а нахера нам собсна использовать крутые дорогие диски, которые умеют быстро читать/писать, если можно использовать говно из жопы, но зато МнОгА!!!!!

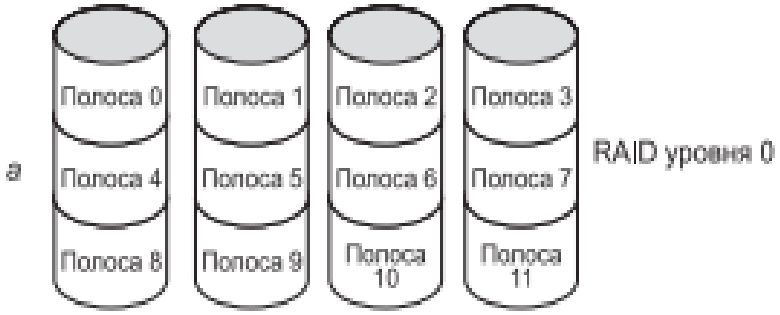
Строго говоря, это способ организации дисков в структуру данных, которую компьютер будет воспринимать как один диск, которая будет круто работать. В зависимости от задач, RAID делят на несколько типов.

2.3.1 RAID0

Ну давайте смотреть.

Дано: $(1 \leq n \leq \infty)$ дисков. Нам нужно о ним разделить информацию. RAID-0 предлагает нам поделить наши диски на полосы по условных k секторов. Теперь давайте разделим нашу информацию на куски, совпадающие с размерами полосы и передадим их RAID контроллеру, он будет записывать каждый такой кусок на следущую свободную полосу, так как каждые две соседние полосы находятся на разных дисках(см. рисунок), то запускать такую запись можно параллельно (на самом деле, пока диски не кончатся). Таким образом скорость работы будет суммой скоростей работы n дисков. Размер не очевиден, в конспектах старшие пишут, что $\min_{n_1,\ldots n_T} S(n_i) * n$, типа как только закончится место на минимально диске мы больше писать не сможем. Но по определению из таненбаума, кажется как только закончится место на минимальном диске RAID контроллер это поймет и распеределит полосы на меньшее количество дисков.

Строго говоря, здесь нет никакой избыточности, поэтому не совсем RAID, но эту структуру относят к ним.

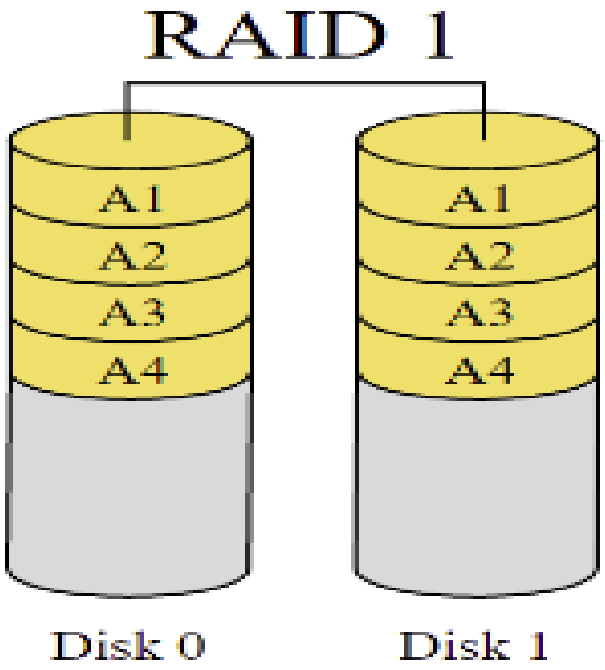


Плюсы: работает лучше с большими запросами, потому что выкладывается структура на полную.

Минусы: работает хуже с небольшими запросами, соответственно. Дисков много, шанс на поломку в n раз больше, а если тебе нужно поменять один сектор, то никакого параллелизма не будет.

2.3.2 RAID1

Ну, вот это уже реальный RAID. Давайте копировать наши данные параллельно во все n дисков. Таким образом скорость записи будет такая же, как и при одном диске, зато скорость счтения, при должном уровне параллелизации(ну мы теперь можем как бы с одного диска n секторов одновременно читать) будет сильно круче.

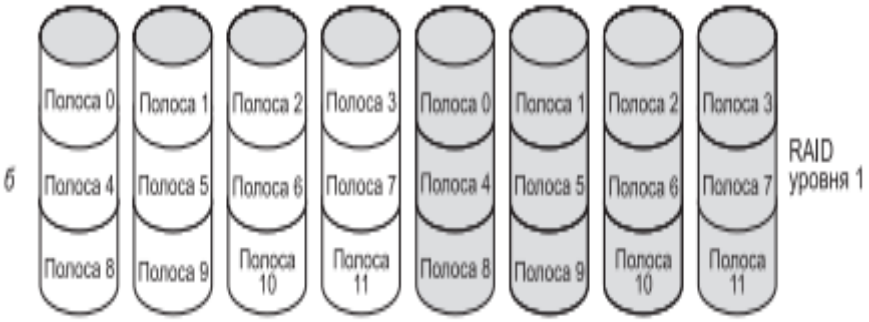


Плюсы: скорость чтения выше, работает пока не сломаются все диски, что круто. Так что, если один из дисков сломался, следует позаботиться о возвращении избыточности.

Минусы: покупая N дисков, на самом деле, мы имеем всего $S(\min_{n_1,\dots n_N})$ полезного простанства.

Вообще, Таненбаум определяет RAID1, как:

Давайте разделим наши диски пополам, и в первой половине организуем RAID0, а во вторую половину будем копировать все, что в первой. Таким образом, записывание будет такое же, так как работает эта вся тема параллельно, а считывание будет быстрее, чем в RAID0 аж до двух раз(потому что мы можем с одного и того же диска считывать одновременно с него и с его копии). Отказоустойчивость тоже норм, сломался диск? -> замениил на его копию. Короче нужно спросить про это у Скакова. //comments: забиваем на это.

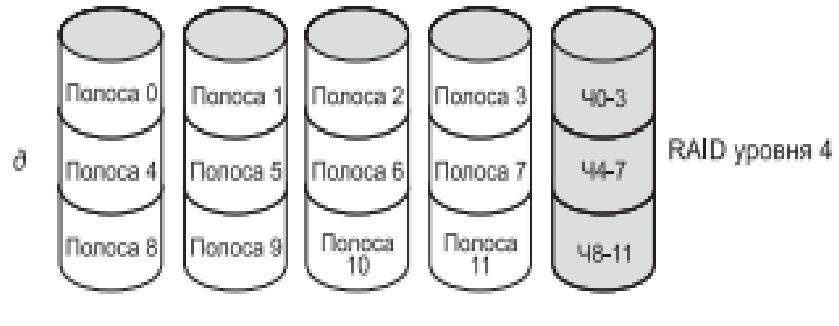


2.3.3 RAID4

Тут все просто, организуем на первых $n-1$ диске RAID0, а на последнем, каждой полоске будем ставить в соответствие битовый ксор всех остальных полосок. По свойствам ксора мы сможем восстанавливать любую полоску, зная все, кроме нее, удобно? Да. Таким образом скорость записи и считывания получаются такими же, как и RAID0(ну без одного диска), но еще и отказоустойчивость значительно повышается.

Плюсы: те же, что и у RAID0 + отказоустойчивость.

Минусы: очень большая нагрузка на диск четности. Любая операция на RAID, вне зависимости от размера запрашиваемых данных обратится к диску четности. Для примера, мы хотим поменять лишь одну полоску, $n-2$ дисков никак не потревожатся, один какой-то диск из первых $n-1$ потревожится, а также еще и диск четности, таким образом нагрзука более или менее распределена между первыми $n-1$ дисками, но совершенно не решена проблема с последним диском, диском четности.



2.3.4 RAID5

Та же идея, что и с RAID4, только нет диска четности, полосы четности равномерно, по кругу распределены среди всех дисков, таким образом нет сильной нагрузки на какой-то конкретный диск.

Плюсы те же, что и у RAID4.

2.3.5 RAID6

То же самое, что и Raid5, только не ксорсумма, а контрольная сумма.

2.4 Флеш память

Ну, шо, все шо не Скаков в мире, то изнашивается и ломается, транзисторы после многочисленных перезаписей приближаются к состоянию недееспособности, одним из путей отказа транзисота является **инжекция горячих носителей** - механизм сбоя, при котором в транзистор внедряется нежелательный электрический заряд, который замыкает его в выключенном или включенном положении навсегда. Ну аниме-челбос из Toshiba придумал как заабузить эти сбои.

Эта очевидная картинка должна была все прояснить(нет). Короче суть в том, что в транзистор встраивается **плавающий затвор**, который мо-

жет заряжаться и разряжаться при подаче высоких напряжений. При подачи напряжения на управляющий затвор, электроны перетекают на плавающий и остаются там. Соответственно ячейка становится заряженой. Внедренный отрицательный заряд увеличивает напряжение, которое нужно транзистору, шобы врубиться. Проверяя включается ли канал на низком напряжении можно сопоставить нули или еденице. Внедренный заряд остается в ячейке памяти, даже при отключении питания, благодаря чему она становится энергонезаисимой. Чтобы убрть заряд с ячейки, подается питание на штуку снизу(каюсь, забыл как ее Скаков назвал) и электроны съебывают с плавающего затвора. Таким образом, ячейка становится незаряженой.

Как происходит чтение: в общем случае, контроллер, когда хочет узнать значение конркетного ячейки(на самом деле он узнает, коненчо, посекторно, но суть понятна), в зависиости от напряжения, сохраненного на плавающем затворе(т.е. есть оно, или его нет) он понимает, на нем ноль или еденица.

Как происходит запись: Тут уже посложнее. Так как, из-за физического устройства памяти, контроллер может обращаться только к блокам(которые чуть больше, чем сектор), он находит блок, в котором содержится ячейка, которую хотим перезаписать. Читает его и сохраняте в буффере, очищает весь этот блок(т.е. убирает заряд), потом заряжает все, что не нужно перезаписать так, как в буфере, а все, что нужно, так как нужно(вау). Таким образом, у вас нет команды "перезаписать есть только команды "очистить"и "записать в чистую область".

Так как физика - хуйня для даунов, ячейки портсятся по чуть-чуть с каждой перезаписью. Контрллер умеет распределять нагрузку между ячейками более или менее равномерно, но все равно каждая ячейка, в зависимости от типа памяти, после некоторого количества раз перестает быть валидной. В зависимости от того, насколько плохо все становится с ячейками, когда мы перезаписываем блоки вводится параметр write amplification.

Можно заметить, что так как теперь ячейка памяти представляет собой не "отрицательный"или "положительный"заряд. А "не заряженнасколько-то заряжен то вот это "насколько-то"можно диффиренцировать. В зависимости от этого, различают типы FLASH памяти:

1. SLC Single Level Cell. Ячейка может принимать значения один/ноль. Перезаписей 10000. Также является самой быстрой, в плане чте-

ния.

2. MLC Multy Level Cell. Ячейка хранит два бита, соответственно принимает 4 значения. Реализуется это увелечичение плавающих затворов. Немного медленней SLC, перезаписей 3000-5000.

3. TLC TiPidor(он не говорил, как это расшифровывается) Level Cell. Ячейка хранит 3 бита, и соответственно принимает 8 значений. Работает ну достатноч медленно. перезаписей 1000-3000.

4. QLC QprumNO(Тоже не говорил) Level Cell. Ячейка хранит 4 бита, принимает 16 значений. Работает супер медленно. перезаписей 16. Пока не существует, но Цукенберг просит, потому что дешево хранить редкоперезаписываемую информацию. ХОМЯЧКАМ, ВРОДЕ ВАС НЕ НУЖНО.

TRIM Ну, хуе мое. Существует такая команда, как TRIM. Ее обязаны поддерживать: ваша ОС, контроллер ССД и все, что между ними(РАИД контроллер, к примеру). Что она делает? Она говорт контроллеру памяти, что данные блоки теперь очищены. В чем ее отличие, от команды очистить? Она очищает именно на логическом уровне, то есть, если вы спросите значение ячейки у контроллера, на которую была применена команда TRIM, он вам скажет, что она очищена, даже если она заряжена. Пока контроллер не занят, он старается реально очистить все, что было затриммено. Это позволяет немного ускорить запись(не обязательно запоминать и перезаписывать ячейки из блока, если они все затриммены). А также, позволяет ускорить очищение, потому что, нам, по сути, нет разницы, реально ли там нули, или контроллер так считает, потому что общаться мы умеем только с контроллеров. Крутота.