# ￼Билет 1.2: Оперативная память: статическая/динамическая, организация

## Оперативная память

Оперативное запоминающее устройство (Random access memory) – вид энергозависимой памяти, позволяющий читать/писать в произвольную ячейку по адресу. В общем случае хранит программы (команды) и данные.

Память состоит из ячеек, каждая из которых хранит некоторое значение и обладает своим уникальным адресом. Ячейки нумеруются последовательно с 0. Каждая физическая ячейка хранит по 1 биту.

## Основные принципы построения памяти

### Простая память с линейной структурой

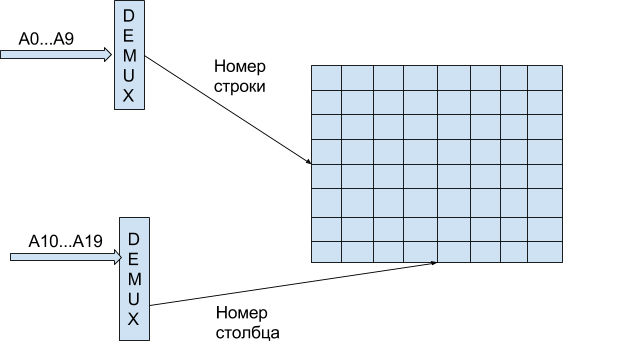
## Снимок экрана 2017-01-06 в 17.27.39.png

А – биты адреса, RW – флаг записи (1 – запись, 0 -- чтение), C – синхронизация, D – данные. Q – выход данных, T -- триггер, хранящий значение ячейки.

Как это работает? При записи, подаем биты адреса, демультиплексор активирует транзистор в нужной ячейке, тем самым соединяя её с линией D. В момент синхронизации записываем значение. При чтении, просто подаем биты адреса, мультиплексор выдаст значение в нужной ячейке.

Однако, оно имеет проблемы – n бит памяти требуют O(2^n) проводников, что не есть хорошо, ибо много параллельных проводников могут привести к искажениям.

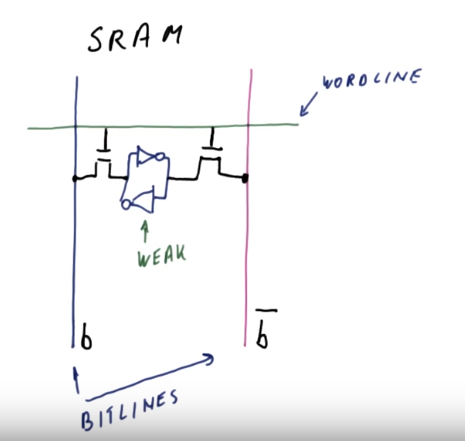
### Уменьшение числа проводников

Будем делать по-другому. Разобьем адрес на две части – A[0..n / 2 - 1] и A[n / 2 .. n - 1]. Теперь, мы можем поставить вместо одного n-битного демультиплексора, два. 

Отлично, тогда у нас будет О(2^(n / 2)) проводников. Уже лучше, не так ли?

Что делать, если мы хотим ещё меньше проводов? Мы можем убрать провода выхода. Почему? Потому что если мы не записываем данные, то читаем. Следовательно, когда используются провода выхода данных, провода входа не используются, и наоборот. Так что давайте использовать для связи с внешним миром одни и те же провода для чтения и записи.

## Статическая память

Используется в основном в качестве небольшой быстрой кэш-памяти.

### Устройство ячейки и принцип действия

Состоит из пары инверторов и двух транзисторов (суммарно 6 транзисторов). Wordline -- линия, открывающая/закрывающая доступ к ячейке, b -- линия, отвечающая за значение в ячейке, not b -- линия, отвечающая за отрицание значения в ячейке.

Чтобы записать значение в ячейку, активируется wordline, тем самым соединяются инверторы и линии данных (b, not b). Затем, на b подается записываемое значение, а на not b -- его отрицание. Если в ячейке хранилось что-либо до этого, то новое значение просто “перезапишет” старое.

После выключения линий b, not b, wordline значение устойчиво хранится. Действует это так: если на одном инверторе на выходе 1, то на выходе следующего получается 0, и так далее, образуется нечто, типа цикла со статическим состоянием. Именно поэтому память является статической, то есть значения устойчиво хранятся, без постоянного обновления, как в DRAM.

Чтобы прочитать значение из ячейки, активируется wordline, тем самым, инверторы подключаются к b и not b. Если значение на линии b больше значения, на линии not b, то в ячейке хранилась 1, иначе -- 0.

Почему существует линия not b? Предположим, что мы убрали транзистор справа и линию not b.

* Тогда, например, если мы хотим записать значение 1 в ячейку, а в ней у нас хранился 0, то нижний инвертор будет мешать нам сделать это, ведь он будет пытаться вернуть значение 0. Итого, чтобы записать 1 мы должны “подавить” 0 более сильным сигналом. Если же у нас будет присутствовать линия not b, мы можем легче подавить старое значение на инверторах, ведь меняем состояние сразу обоих инверторов.
* Если мы хотим прочитать значение из ячейки без линии not b, нам придется “честно” смотреть на значение на линии b, а сигнал может быть недостаточно силен, чтобы отличить 0 от 1.

### Преимущества и недостатки

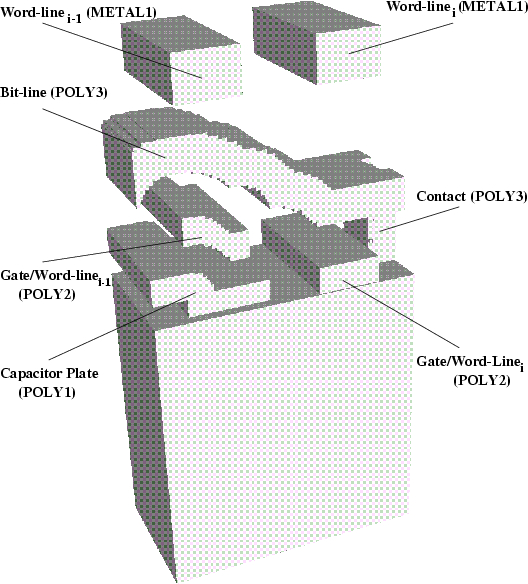
* Данные хранятся всё время, пока подается питание.
* Высокая скорость работы, переключение транзисторов выполняется быстрее, нежели зарядка конденсатора в DRAM.
* Более простая компоновка с другими системами, ведь не требуется система, регенерирующая значения в ячейках.
* Низкая плотность записи – на каждый бит требуется 6 транзисторов.
* Более дорогая, из-за большего количества транзисторов на ячейку.
* Требовательна к энергопотреблению из-за большого количества транзисторов.

## Динамическая память

Используется в качестве основной матрицы ячеек в современных ОЗУ

### Устройство ячейки и принцип действия

Основана на паре транзистор-конденсатор. При чтении из такой ячейки заряд конденсатора уходит, поэтому после чтения данных их копию нужно записывать обратно. Также, заряд в конденсаторах имеет тенденцию утекать/исчезать, следовательно, каждые несколько миллисекунд необходимо его восстанавливать. Об обновлении данных, согласно Таненбауму, должны заботиться некоторая внешняя система (в современных системах -- контроллер памяти, который интегрирован в процессор), поэтому само устройство и сопряжение такой памяти сложнее, нежели SRAM.

Конденсатор в современных DRAM “утоплен” в подложку, то есть находится под транзистором, что позволяет экономить ещё больше места. (картинка справа).

Синяя (вертикальная) -- линия данных b, зеленая (горизонтальная) -- линия управления wordline.

Чтобы записать значение в ячейку, активируется wordline, тем самым, соединяется конденсатор и b. Затем на b подается необходимое значение и деактивируется wordline, отключая конденсатор от линии данных.

После отключения wordline и b, заряд конденсатора постепенно утекает, поэтому периодически значение в ячейках надо восстанавливать.

Чтобы прочитать значение из ячейки, активируется wordline, тем самым соединяется конденсатор и линия данных. Значение из конденсатора (заряд) передается в b, и конденсатор остается пуст, то есть чтение разрушает данные в ячейке, поэтому их необходимо восстанавливать.

### Преимущества и недостатки

DRAM обладает более высокой плотностью записи, так как на 1 бит используется 1 конденсатор и 1 транзистор на бит. Поэтому основной массив памяти в современных модулях ОЗУ является как раз динамической памятью. Однако, DRAM обладает гораздо более низкой скоростью чтения, потому что сигнал, выдаваемый конденсатором в линию данных слабый, значит, необходимо время для его усиления, чтобы с ним работать дальше.

* Высокая плотность записи – на 1 бит нужен всего 1 транзистор и 1 конденсатор
* Низкая цена, по сравнению с SRAM
* Заряд с течением времени теряется, следовательно, необходим процесс его регенерации, коим занимается контроллер памяти
* Низкая скорость доступа (зарядка/разрядка конденсатора происходит медленнее переключения триггеров)
* После чтения данные уничтожаются, нужно помещать их обратно.
* Более сложная компоновка, вследствие предыдущих недостатков.

## Общее устройство памяти (однопортовой)

* Row addr -- адрес строки
* Column addr -- адрес столбца
* Row decoder -- устанавливает в 1 управляющую линию с заданным адресом.
* Sense amplifier -- усилитель сигнала. Усиливает слабый сигнал на линии данных до приемлемого уровня
* Row buffer -- буферная (интерфейсная) строка.
* Column decoder -- мультиплексор, возвращает значение ячейки по её адресу в строке.
* На каждом пересечении расположена ячейка памяти

### Чтение

Пусть пришел запрос на чтение значения из ячейки с адресом A. Как было сказано выше, первая часть адреса отправляется на вход в row decoder, вторая -- в column decoder. Происходит активация управляющей линии строки, содержащей нашу ячейку, значения проходят через усилитель, записываются в буферную строку. Затем, происходит чтение нужного столбца из буферной строки.

### Запись

Пусть пришёл запрос на запись в ячейку A. Активируется управляющая линия нужной строки, значения через усилитель попадают в буферную строку. Затем, происходит запись значения в нужный столбец буферной строки и запись строки обратно в матрицу. Запись также происходит через column decoder.

### Усилитель

Зачем нужен усилитель? Как в статической, так и в динамической ячейке, при соединении с линией данных (довольно длинным проводником, относительно размеров самой ячейки), напряжение на ней будет меняться, но недостаточно сильно, чтобы дальше с ним работать. Усилитель же берет значения на линии и “выравнивает” их в нужную сторону до приемлемых.

### Буферная (интерфейсная) строка

При матричном устройстве памяти, мы не имеем возможности прочитать или записать напрямую в конкретную ячейку, так как при активации управляющей линии строки, мы работаем сразу со всеми ячейками строки.

В случае DRAM при чтении значения строки в основной матрице разрушаются, поэтому после работы со строкой, они записываются обратно.

В случае SRAM значения не разрушаются, но, так как мы не имеем возможности записать только в конкретную ячейку, мы должны запоминать предыдущие значения в строке.

### Регенерация значений в DRAM

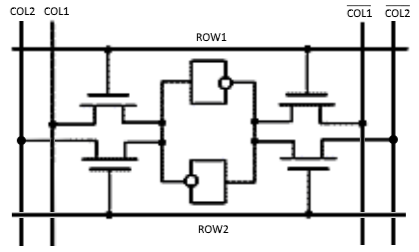
И чтение, и запись в DRAM на самом деле являются парой операций чтение-запись, за счет того, что чтение из ячейки динамической памяти разрушает данные. Это одна из причин, почему динамическая память медленнее, чем статическая.

Таким образом, при любой операции со строкой динамической памяти, значения в ней регенерируются. Но что делать со строками, к которым нет обращений? Заставлять заниматься такой ерундой процессор -- не комильфо. Ровно так же, мы не можем положиться в этом деле на программную часть -- операционную систему из-за существования кэша процессора. Может случиться ситуация, что строка, которую нужно регенерировать, находится в кэше, и тогда эта строка не будет обновлена в ОЗУ.

//Спросить у скакова, а как же на самом деле регенерируются строки, я чето не очень понял

## Многопортовая память

Выше было дано описание однопортовой памяти, которая имеет существенный недостаток -- невозможно работать одновременно более, чем с одной строкой. Этот недостаток устранен в многопортовой памяти. *(Ниже приведена схема двухпортовой статической ячейки памяти)*

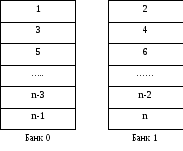
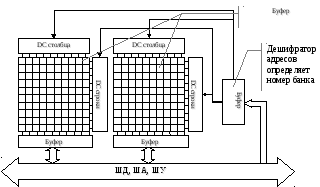
В многопортовой памяти каждая ячейка по-прежнему имеет 1 блок для запоминания значений (транзистор-конденсатор или D-триггер), но имеет несколько комплектов управляющих транзисторов. 

Таким образом становится возможной работа с разными строками в один и тот же момент времени. Наиболее часто ныне используется двухпортовая память, то есть каждая ячейка имеет 2 комплекта управляющих транзисторов.

## Многобанковая память

Банк памяти -- набор страниц (строк) памяти, находящихся в одной матрице. Соответственно, разные матрицы памяти есть разные банки.

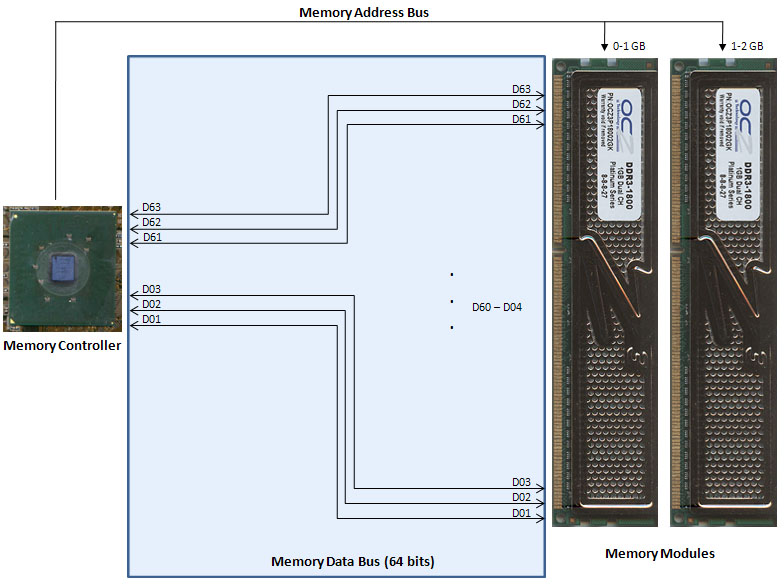
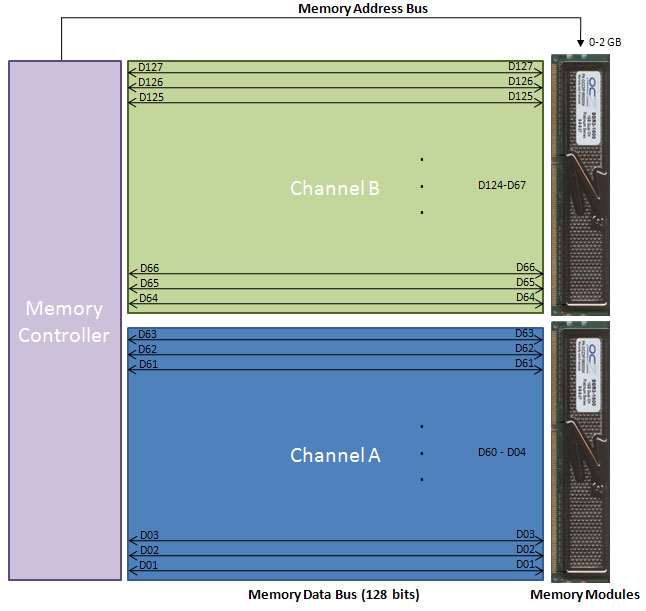
Зачем это надо и почему бы не сделать один большой банк? Если у нас есть много банков поменьше, то мы можем повысить быстродействие. Например, пока идет чтение данных из матрицы в буферную строку в одном банке, мы можем работать с данными в другом банке. Или когда один банк находится на регенерации. (Этот момент связан с регенерацией, будет уточнен у скэкова).



Выше -- пример двухбанкового устройства одного модуля памяти (то есть банки имеют общие управляющие структуры и общий выход во внешний мир). Если идти по ячейкам последовательно, то они будут попадать то в один банк, то в другой. Следовательно, пока идет чтение/запись в одном банке, в другом банке может идти регенерация. Такая схема называется доступом с чередованием банков.

## Многоканальная память

*Верхняя картинка -- одноканальная память, нижняя -- двухканальная.*

****

В чем проблема одноканальной памяти? В том, что память, работающая в одноканальном режиме может передать только 64 бита данных за раз. А в условиях возрастающего быстродействия современных систем, шина памяти всё чаще становится узким местом.

Решение проблемы заключается в следующем: увеличить разрядность шины в n раз и поставить количество модулей памяти, кратное n. Тогда, теоретически, пропускная способность шины будет увеличена в n раз по сравнению с одноканальной архитектурой того же объёма.

При этом необходимо, чтобы параметры модулей памяти в одном канале совпадали (работали на одинаковых частотах, имели одинаковую конфигурацию)

## FAQ

### Почему ячейки хранятся матрично, а не в линию?

Если ячейки будут расположены линейно, то будет много параллельных проводников, что не есть хорошо. (Пункт “Простая память с линейной структурой”)

Если по одному проводнику течет ток в каком-либо направлении, то этот ток порождает вокруг проводника магнитное поле. Вроде бы, (но это не точно), магнитное поле может порождать движение зарядов в соседнем проводнике, то есть, приводить к искажению сигнала (замена 0 на 1, например).

### Почему номера строк и столбцов лучше передавать последовательно, а не параллельно?

Так как память ныне устроена матрично, то передавать адрес ячейки целиком нет смысла, ведь всё равно сначала будет некоторое ненулевое время “открываться” нужная строка памяти, а только потом будет выбираться нужный столбец.

Что у тебя происходит, когда ты пытаешься прочитать ячейку? У тебя происходит открытие определенной строки, а для этого нужно время, то есть номер столбца будет использован не сразу. Так и смысл передавать его сразу, если можно передать сначала одно, потом другое, тем самым, уменьшив шину адреса?

При этом, если мы пытаемся построить память с одновременной передачей строки и столбца, то у нас получится опять много входящих параллельных проводов.

### Что схожего между однопортовой и двухпортовой памятью?

Очевидно, строение ячейки почти не меняется, смотрите схемы выше.

### Может ли DRAM быть двухпортовой? Зачем она нужна? И почему не используют двухпортовую архитектуру в обычной ОЗУ?

Да, например, видеопамять, VRAM. Проблема однопортовой видеопамяти в том, что данные оттуда читаются постоянно с частотой обновления картинки на экране. Однако, параллельно этому необходимо ещё и писать данные в эту же самую память.

Почему не делают обычную ОЗУ двухпортовой? Потому что это дорого. (Хотя, есть всякие DPRAM).

### Когда бывает выгодно отправлять адрес строки и столбца одновременно?

В случае, когда память синхронная. Тогда, к приходу следующего тактового импульса, сигналы успеют стабилизироваться. (Ведь переключение сигналов происходит не мгновенно). (Вроде ответ такой, мб спросить скакова)