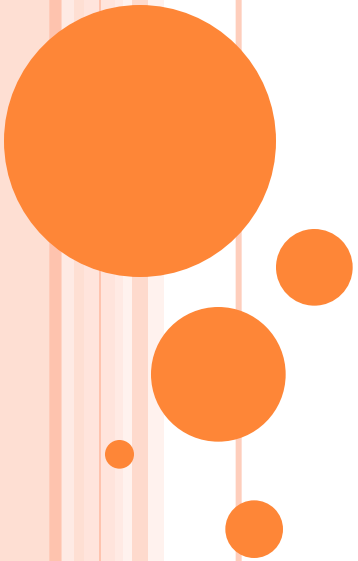
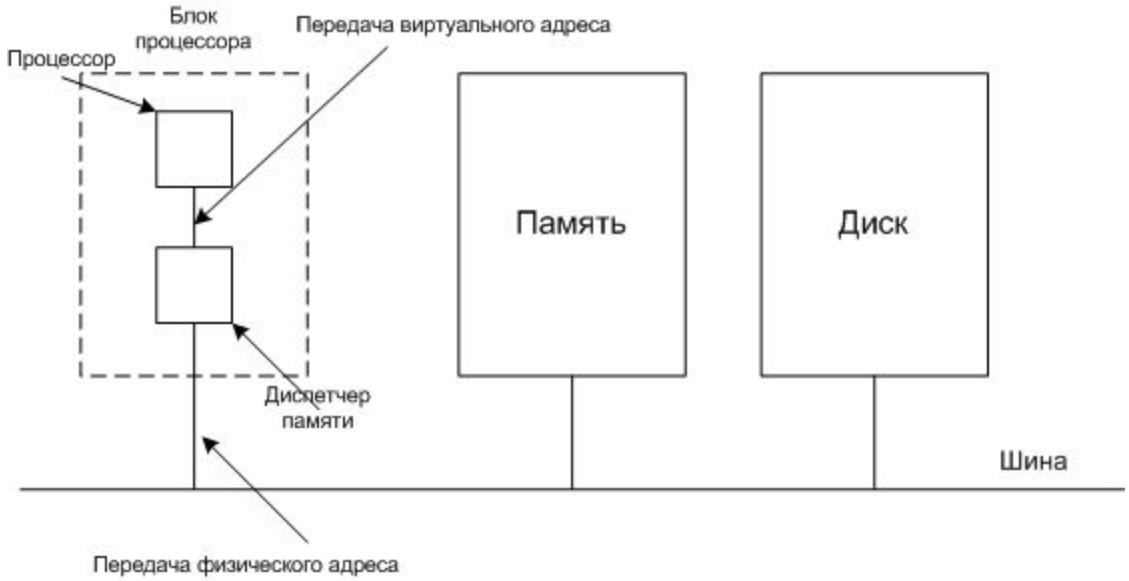
# Управление памятью



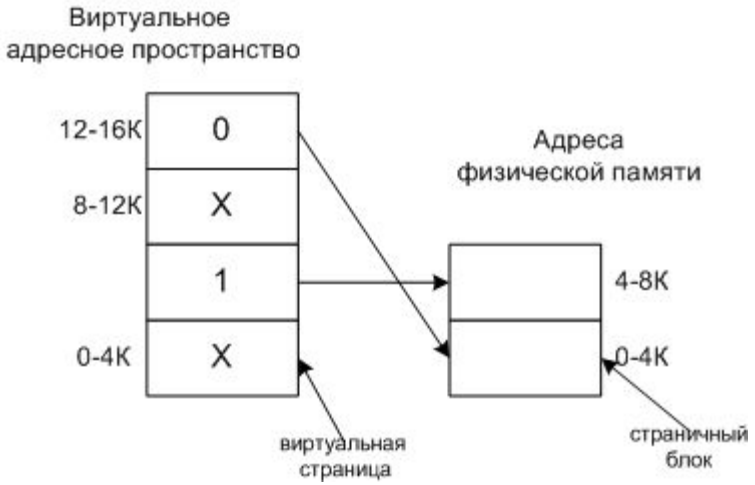
# ВИРТУАЛЬНАЯ ПАМЯТЬ

- Основная идея заключается в разбиении программы на части, и в память эти части загружаются по очереди.
- Программа при этом общается с виртуальной памятью, а не с физической.



# СТРАНИЧНАЯ ОРГАНИЗАЦИЯ ПАМЯТИ

- **Страницы** - это части, на которые разбивается пространство виртуальных адресов.
- **Страничные блоки** - единицы физической памяти.
- Страницы всегда имеют фиксированный размер. Передача данных между ОЗУ и диском всегда происходит в страницах.

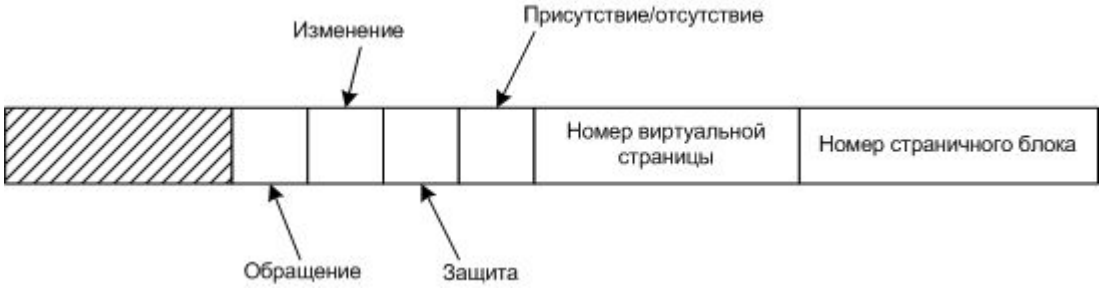



X - обозначает не отображаемую страницу в физической памяти.

# СТРАНИЧНАЯ ОРГАНИЗАЦИЯ ПАМЯТИ

- **Страничное прерывание** - происходит, если процесс обратился к странице, которая не загружена в ОЗУ (т. е. X). Процессор передается другому процессу, и параллельно страница загружается в память.
- **Таблица страниц** - используется для хранения соответствия адресов виртуальной страницы и страничного блока.
- Таблица может быть размещена:
  - в аппаратных регистрах (преимущество: более высокое быстродействие, недостаток - стоимость)
  - в ОЗУ

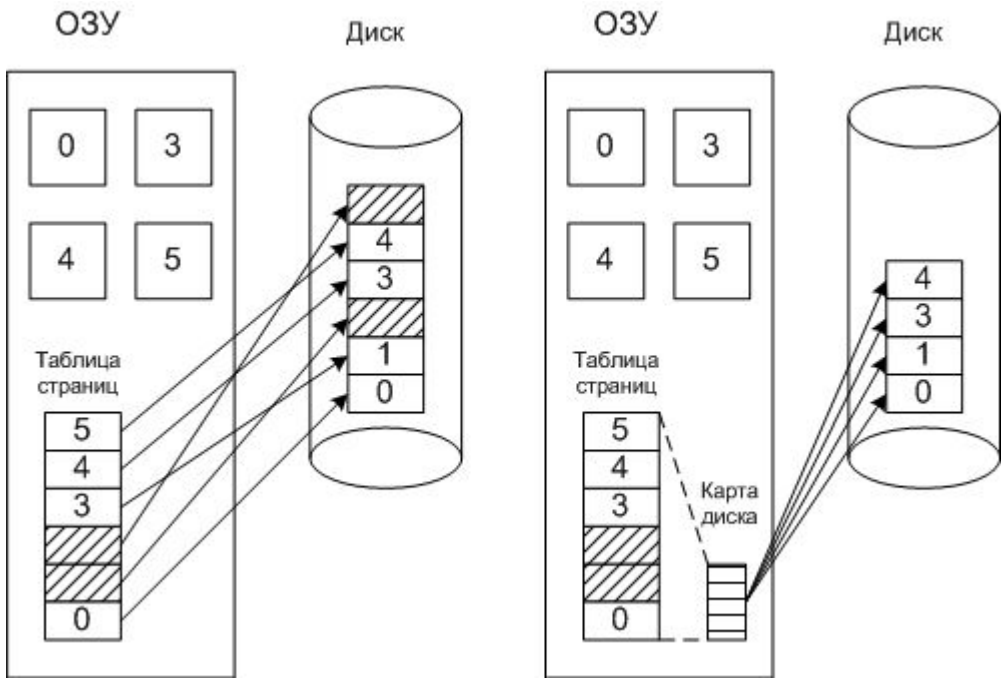
# СТРАНИЧНАЯ ОРГАНИЗАЦИЯ ПАМЯТИ




- Присутствие/отсутствие - загружена или незагружена в память
- Защита - виды доступа, например, чтение/запись.
- Изменение - изменилась ли страница, если да то при выгрузке записывается на диск, если нет, просто уничтожается.
- Обращение - было ли обращение к странице, если нет, то это лучший кандидат на освобождение памяти.
- *Информация о адресе страницы когда она хранится на диске, в таблице не размещается.*
- Для ускорения доступа к страницам в диспетчере памяти создают **буфер быстрого преобразования адрес**а, в котором хранится информация о наиболее часто используемых страниц.
- *Страничная организация памяти используется, и в UNIX, и в Windows.*

# ХРАНЕНИЕ СТРАНИЧНОЙ ПАМЯТИ НА ДИСКЕ

- После запуска процесса он занимает определенную память, на диске сразу ему выделяется такое же пространство. Поэтому файл подкачки должен быть не меньше памяти. А в случае нехватки памяти даже больше. Как только процесс завершится, он освободит память и место на диске.
- На диске всегда есть дубликат страницы, которая находится в памяти.
- Этот механизм наиболее простой.



# АЛГОРИТМЫ ЗАМЕЩЕНИЯ СТРАНИЦ

- Идеальный алгоритм заключается в том, что бы выгружать ту страницу, которая будет запрошена позже всех. Этот алгоритм не осуществим, т.к. нельзя знать какую страницу, когда запросят. Можно лишь набрать статистику использования.

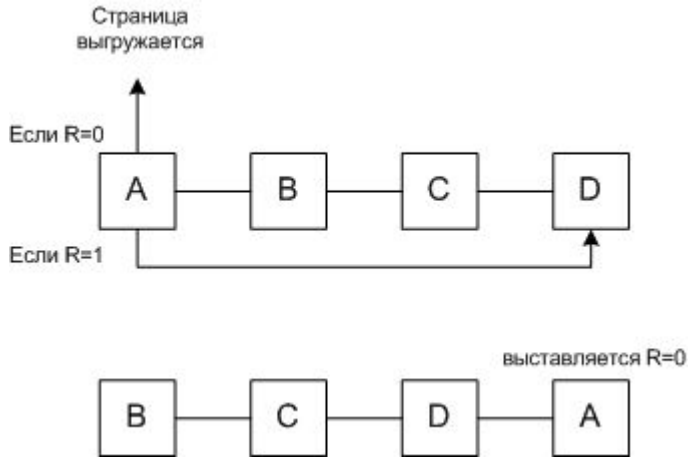
# АЛГОРИТМ NRU (NOT RECENTLY USED)

Лучше выгрузить измененную страницу, к которой не было обращений по крайней мере в течение одного тика системных часов, чем стереть часто используемую страницу.

- Используются биты обращения (R-Referenced) и изменения (M-Modified) в таблице страниц.
- При обращении бит R выставляется в 1, через некоторое время ОС не переведет его в 0.
- M переводится в 0, только после записи на диск.
- Благодаря этим битам можно получить 4-ре класса страниц:
  - не было обращений и изменений (R=0, M=0)
  - не было обращений, было изменение (R=0, M=1)
  - было обращение, не было изменений (R=1, M=0)
  - было обращений и изменений (R=1, M=1)

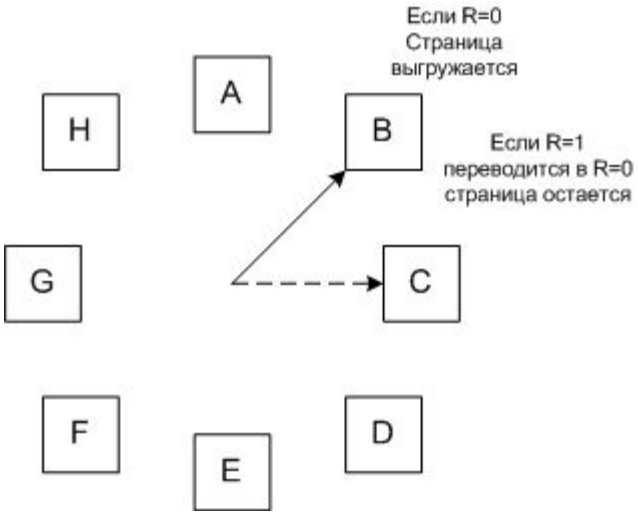
# АЛГОРИТМЫ

- FIFO (первая прибыла – первая выгружена)
- Алгоритм «вторая попытка»
- Подобен FIFO, но если R=1, то страница переводится в конец очереди, если R=0, то страница выгружается. Часто используемая страница никогда не покинет память, но приходится часто перемещать страницы по списку.



# АЛГОРИТМ «ЧАСЫ»

- Чтобы избежать перемещения страниц по списку, можно использовать указатель, который перемещается по списку.

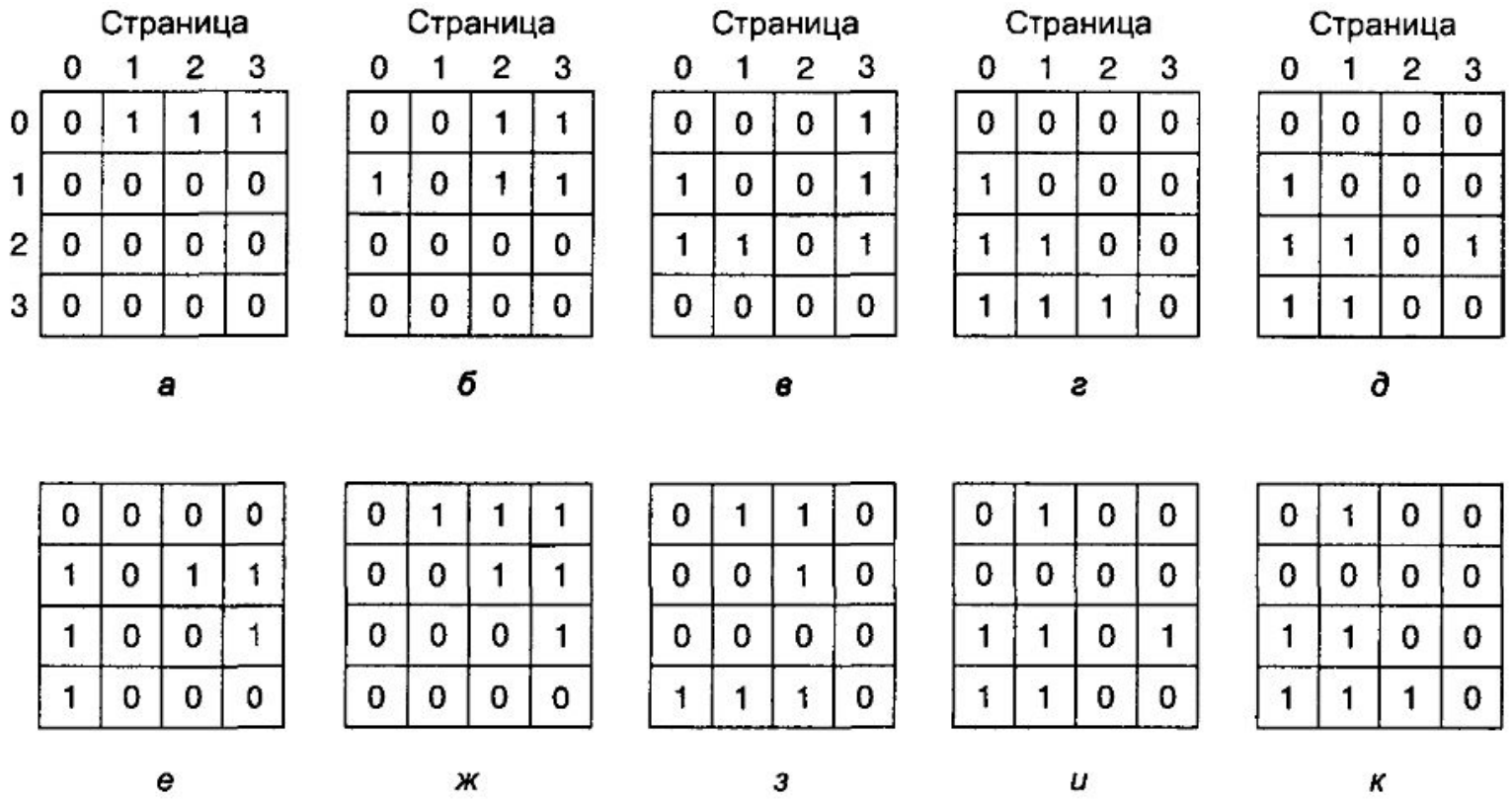


# АЛГОРИТМ LRU (LEAST RECENTLY USED)

- Поддерживать список, в котором выстраивать страницы по количеству использования. Эта реализация очень дорога.
- В таблице страниц добавляется запись - счетчик обращений к странице. Чем меньше значение счетчика, тем реже она использовалась.
- Машина с n страничными блоками. n X n – матрица. При обращении к блоку k, аппаратура сначала присваивает всем битам строки k значение 1, затем всем биты столбца k присваивает 0.
- В любой момент времени строка, двоичное значение которой наименьшее, является не использовавшейся дольше всего.

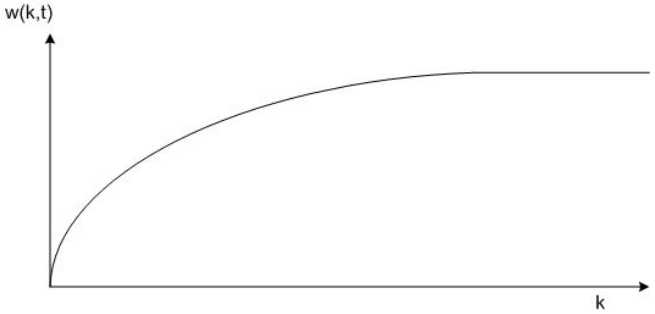
# LRU

Обращения к страницам: 0 1 2 3 2 1 0 3 2 3

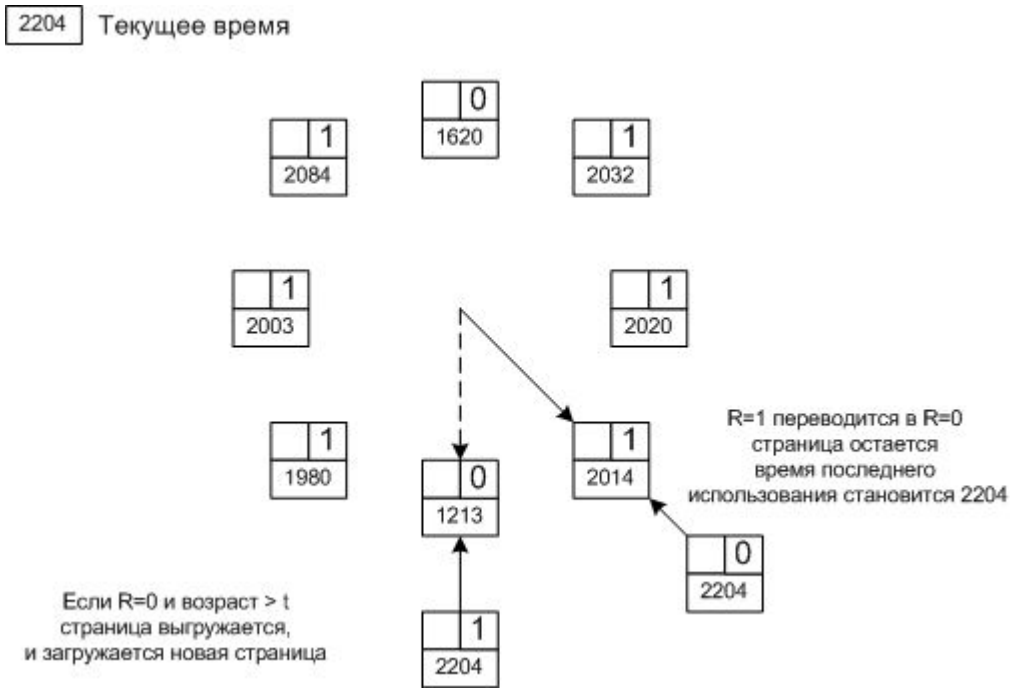


# АЛГОРИТМ «РАБОЧИЙ НАБОР»

- **Замещение страниц по запросу** - когда страницы загружаются по требованию, а не заранее, т.е. процесс прерывается и ждет загрузки страницы.
- **Буксование** - когда каждую следующую страницу приходится процессу загружать в память.
- Чтобы не происходило частых прерываний, желательно чтобы часто запрашиваемые страницы загружались заранее, а остальные подгружались по необходимости.
- **Рабочий набор** - множество страниц (к), которое процесс использовал до момента времени (t). Т.е. можно записать функцию w(k,t).




- Т.е. рабочий набор выходит в насыщение, значение w(k,t) в режиме насыщения может служить для рабочего набора, который необходимо загружать до запуска процесса.
- Алгоритм заключается в том, чтобы определить рабочий набор, найти и выгрузить страницу, которая не входит в рабочий набор.
- Этот алгоритм можно реализовать, записывая, при каждом обращении к памяти, номер страницы в специальный сдвигающийся регистр, затем удалялись бы дублирующие страницы. Но это дорого.
- В принципе можно использовать множество страниц, к которым обращался процесс за последние **t** секунд.
- **Текущее виртуальное время (Tv)** - время работы процессора, которое реально использовал процесс.
- **Время последнего использования (Told)** - текущее время при R=1, т.е. все страницы проверяются на R=1, и если да то текущее время записывается в это поле.
- Теперь можно вычислить возраст страницы (не обновления) **Tv-Told,** и сравнить с **t**, если больше, то страница не входит в рабочий набор, и страницу можно выгружать.
- Получается три варианта:
  - если R=1, то текущее время запоминается в поле время последнего использования
  - если R=0 и возраст > t, то страница удаляется
  - если R=0 и возраст =< t, то эта страница входит в рабочий набор

# АЛГОРИТМ WSCLOCK

- Используются битов R и M, а также время последнего использования.