Лекция 5. Управление памятью

Память является важнейшим ресурсом, требующим тщательного управления со стороны мультипрограммной операционной системы.

- *Функциями ОС по управлению памятью являются*:

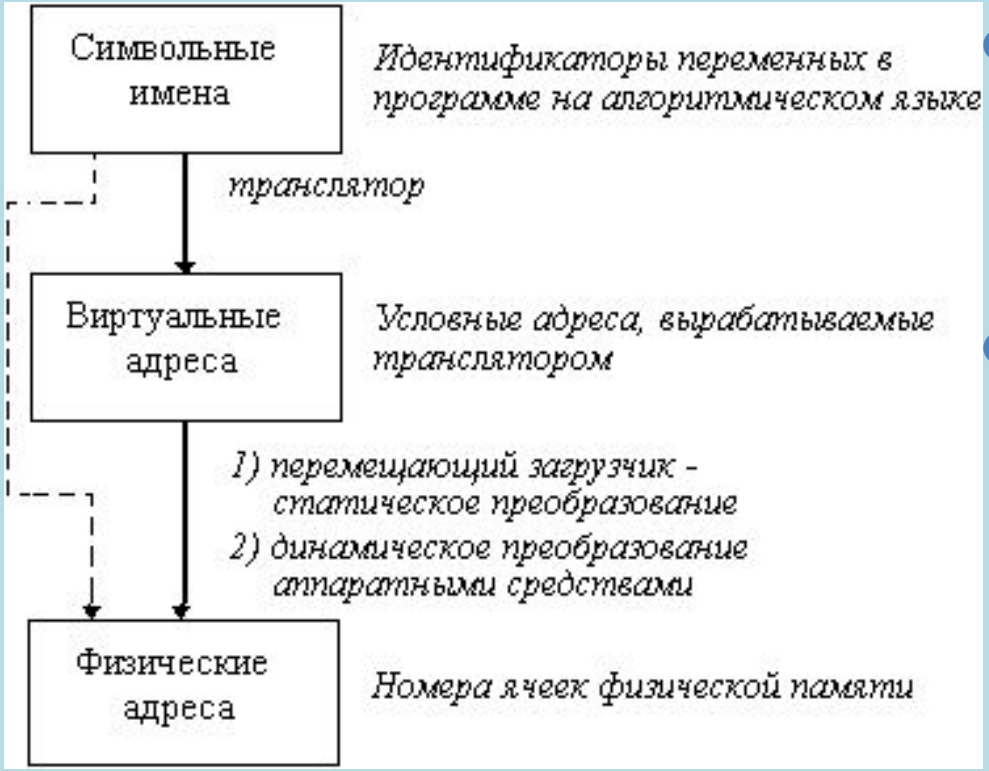
✔ отображение адресов программы на конкретную область физической памяти

✔ распределение памяти между конкурирующими процессами и защита адресных пространств процессов

✔ выгрузка процессов на диск, когда в оперативной памяти недостаточно места для всех процессов

✔ учет свободной и занятой памяти

- Одна из функций управления памятью является преобразование адресных пространств.
- Адреса, с которыми имеет дело менеджер памяти, бывают логическими (виртуальные для систем с виртуальной памятью) и физическими



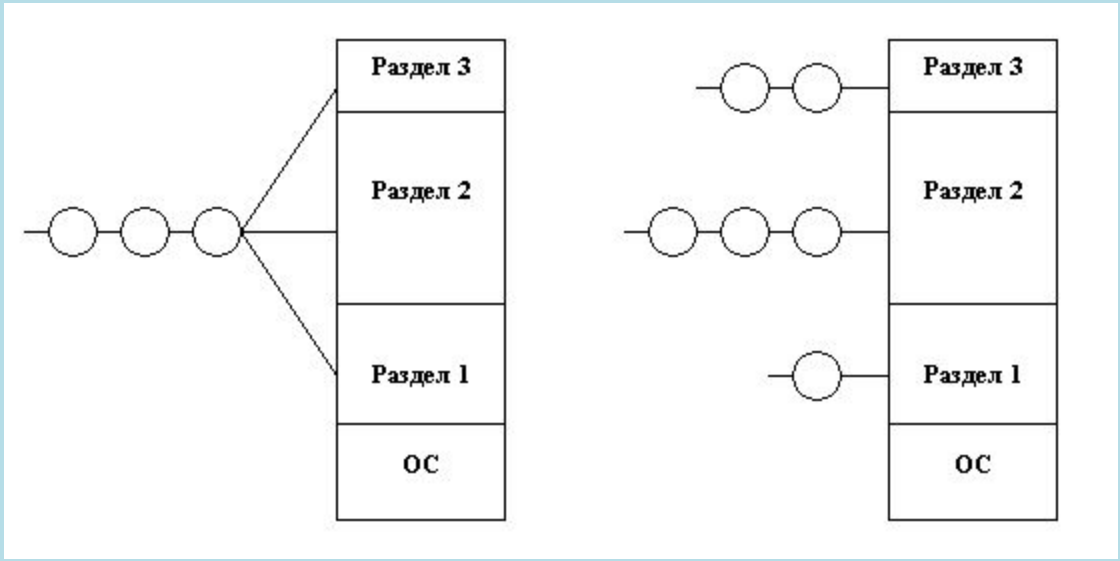
- Пользовательская программа имеет дело с логическими адресами, которые являются результатом трансляции символьных имен программы
- Логические адреса обычно образуются на этапе создания загрузочного модуля (компоновки программы)

- Набор адресов, сгенерированный программой, называют *логическим (виртуальным) адресным пространством*, которому соответствует *физическое адресное пространство*.
- Максимальный размер логического адресного пространства обычно определяется разрядностью процессора (например, 2**32) и обычно превышает размер физического адресного пространства
- Связывание логического адреса, порожденного оператором программы, с физическим адресом должно быть осуществлено до начала выполнения оператора или в момент его выполнения
- Обычно программа проходит нескольких шагов: текст на языке программирования, объектный модуль, загрузочный модуль, бинарный образ в памяти
- В каждом конкретном случае адреса могут быть представлены различными способами Адреса в исходных текстах обычно символические. Компилятор связывает символические адреса с перемещаемыми адресами. Компоновщик связывают перемещаемые адреса с виртуальными адресами. Каждое связывание - отображение одного адресного пространства в другое


Методы распределения памяти
Без использования внешней памяти
С использованием внешней памяти
Фиксированными разделами
Динамическими разделами
Перемещаемыми разделами
Страничное распределение
Сегментное распределение
Сегментно-страничное распределение

Схема с фиксированными разделами

- Каждый раздел может иметь свою очередь или может существовать глобальная очередь для всех разделов

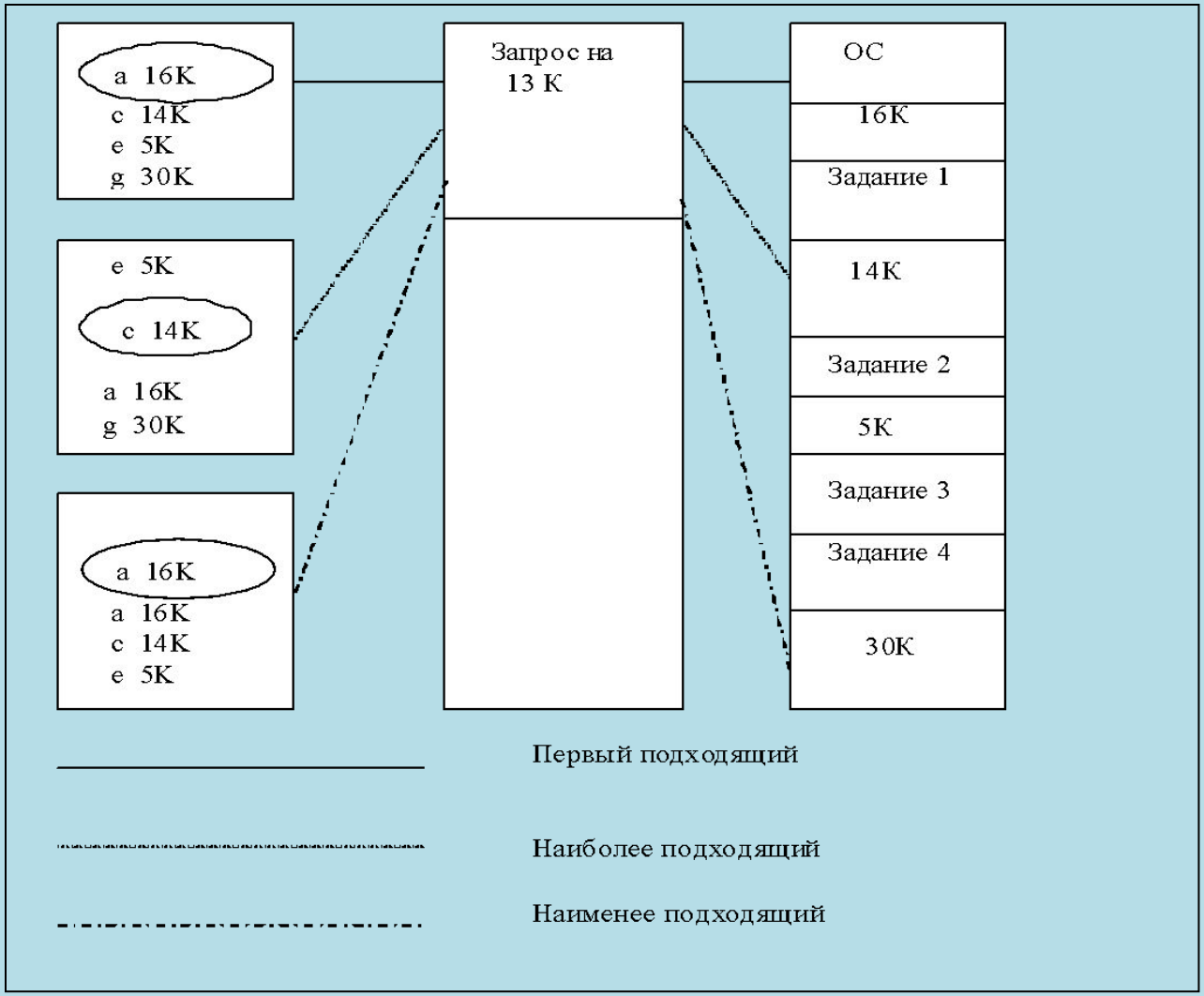


- Эта схема была реализована в IBM OS/360 (MFT) и в DEC RSX-11
- Подсистема управления памятью оценивает размер программы

- Выбирает подходящий раздел, производит загрузку программы и настройку адресов

Стратегии размещения информации в оперативной памяти.

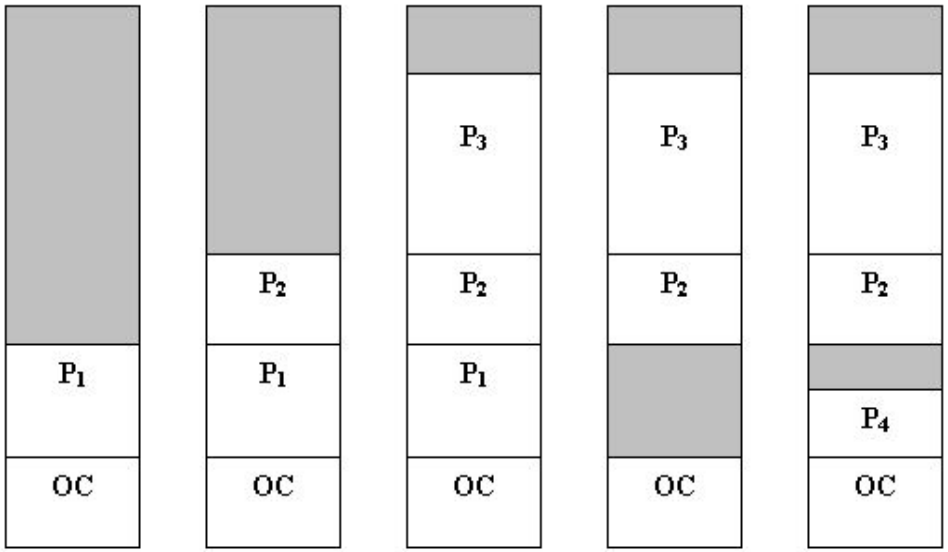
1. Стратегия НАИБОЛЕЕ подходящего;
2. Стратегия ПЕРВОГО подходящего;
3. Стратегия НАИМЕНЕЕ подходящего.



- Очевидным недостатком схемы является то, что число одновременно выполняемых процессов ограничено числом разделов.
- Другим существенным недостатком является то, что предлагаемая схема сильно страдает от *внутренней фрагментации*, потери памяти, не используемой ни одним процессом
- Фрагментация возникает потому, что процесс не полностью занимает выделенный ему раздел или вследствие не использования некоторых разделов, которые слишком малы для выполняемых пользовательских программ

- Частный случай схемы с фиксированными разделами используется в работе менеджера памяти однозадачной ОС
- В памяти размещается один пользовательский процесс
- Остается определить, где располагается пользовательская программа по отношению к ОС - сверху, снизу или посередине
- При этом часть ОС может располагаться в *ROM (read-only-memory)* (например, BIOS, драйверы устройств)
- Главный фактор, влияющий на это решение - расположение вектора прерываний, который обычно локализован в нижней части памяти, поэтому ОС также размещают в нижней
- Примером такой организации может служить ОС MS-DOS
- Чтобы пользовательская программа не портила код ОС, требуется защита ОС, которая может быть организована при помощи одного граничного регистра, содержащего адрес границы ОС

Распределение памяти разделами переменной

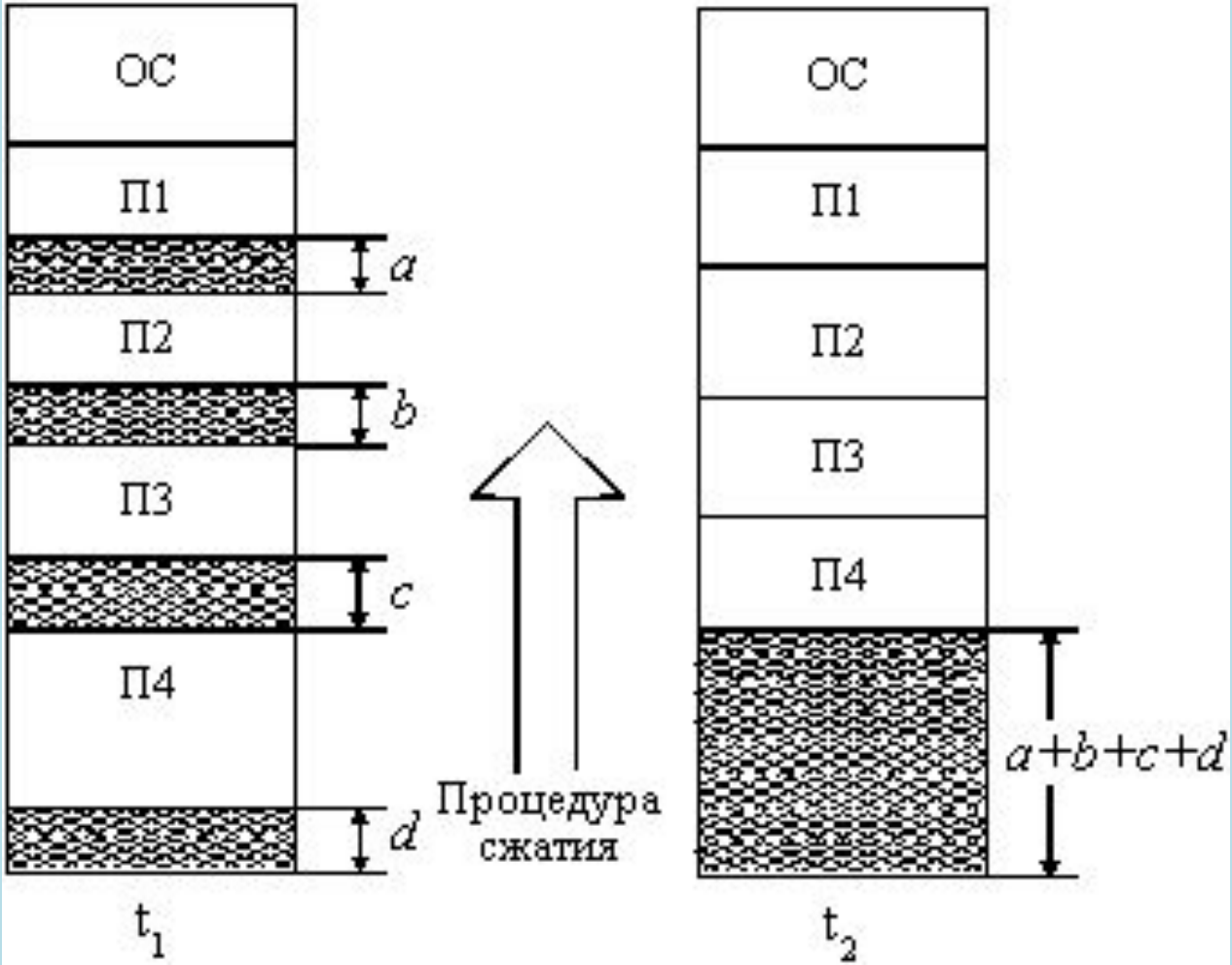


- По истечении некоторого времени память представляет собой набор занятых и свободных участков
- Смежные свободные участки могут быть объединены в один.
- Связывание адресов может быть на этапах загрузки и выполнения

- Типовой цикл работы менеджера:

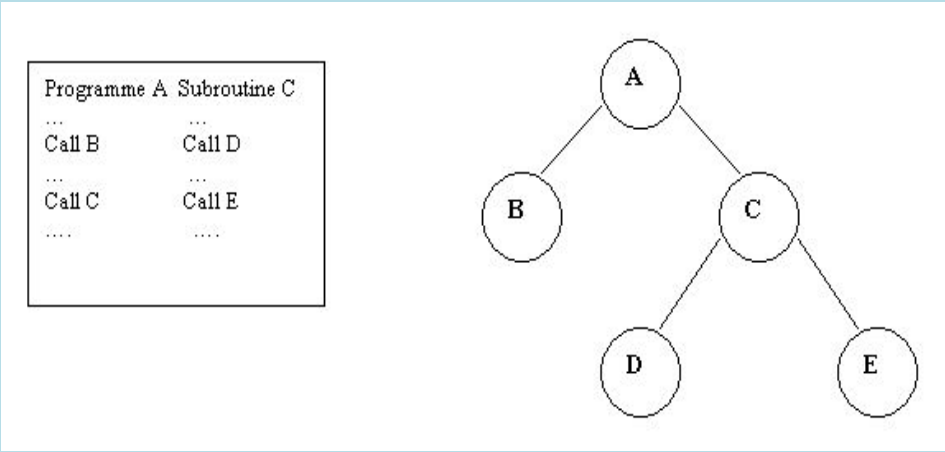
✔ анализ запроса на выделение свободного участка (раздела);
✔ выборка его среди имеющихся в соответствие с одной из стратегий
✔ загрузка процесса в выбранный раздел
✔ внесение изменений в таблицы свободных и занятых областей

Перемещаемые разделы



Распределение памяти с использованием дискового пространства

- Давно существует проблема размещения в памяти программ, размер которых превышает размер доступной ОП
- Один из вариантов ее решения - организация структур с перекрытием (оверлеями)



- Предполагалось активное участие программиста в процессе сегментации и загрузки программы
- Позже было предложено переложить решение проблемы на компьютер
- Одним из основных усовершенствований архитектуры стало появление *виртуальной памяти* (*virtual memory*). Впервые была реализована в 1959 г. на компьютере Атлас, разработанном в Манчестерском университете

Виртуальная память - это совокупность программно-аппаратных средств, позволяющих пользователям писать программы, размер которых превосходит имеющуюся оперативную память; для этого виртуальная память решает следующие задачи:

- размещает данные в запоминающих устройствах разного типа, например, часть программы в оперативной памяти, а часть на диске;
- перемещает по мере необходимости данные между запоминающими устройствами разного типа, например, подгружает нужную часть программы с диска в оперативную память;
- преобразует виртуальные адреса в физические.

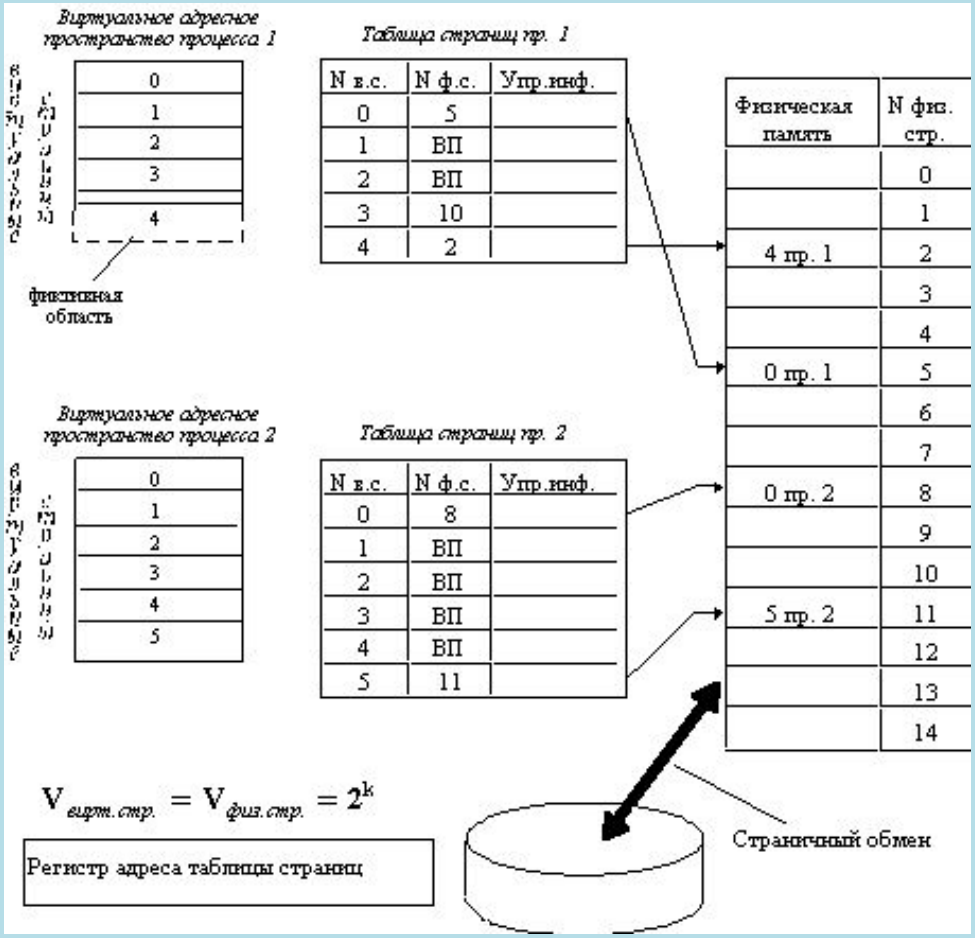
Все эти действия выполняются *автоматически*, без участия программиста, то есть механизм виртуальной памяти является прозрачным по отношению к пользователю.

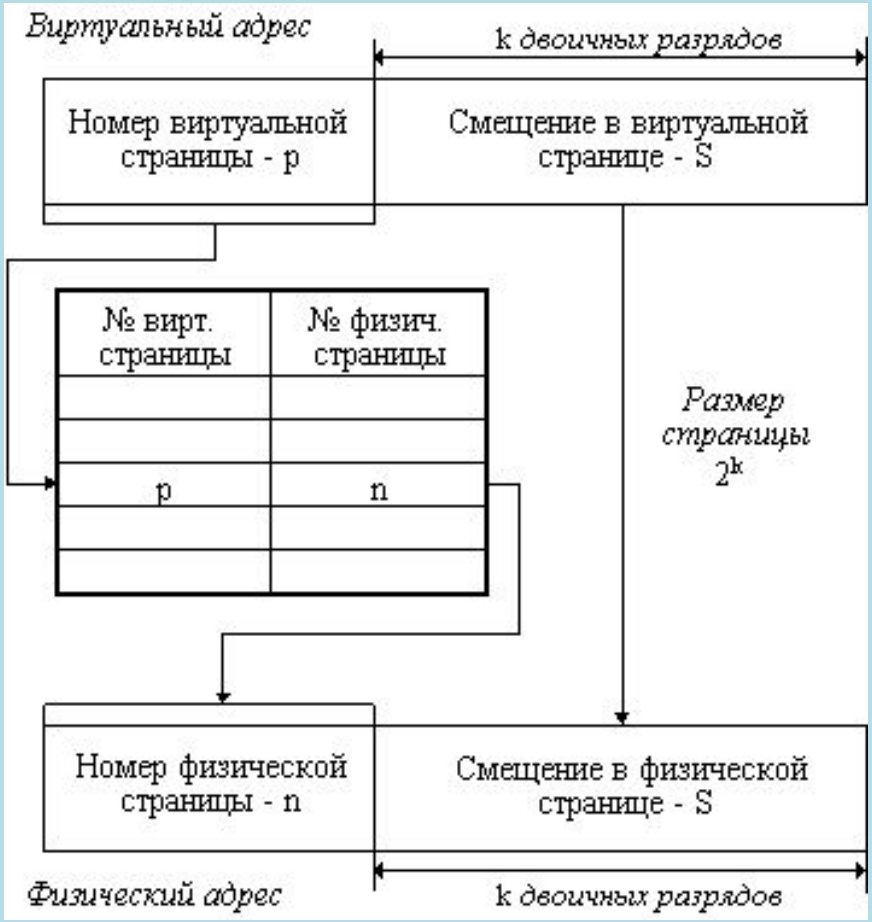
- Виртуальная память позволяет адресовать пространство, гораздо большее, чем емкость физической памяти конкретной вычислительной машины
- В соответствии с принципом локальности для реальных программ обычно нет необходимости в помещении их в физическую память целиком
- Возможность выполнения программы, находящейся в памяти лишь частично, обеспечивает ряд преимуществ

1. Программа не ограничена размером ОП; упрощается разработка программ, поскольку можно задействовать большие виртуальные пространства, не заботясь о размере используемой памяти
2. Возможность частичного помещения программы (процесса) в память и гибкого перераспределения памяти между программами позволяет разместить в памяти больше программ, что увеличивает загрузку процессора и пропускную способность системы
3. Объем ввода-вывода для выгрузки части программы на диск может быть меньше, чем при свопинге, и поэтому каждая программа будет работать быстрее

Страничное распределение памяти

- В наиболее простом и наиболее часто используемом случае страничной ВП она (и ОП) представляются состоящими из наборов блоков или страниц одинакового размера
- Виртуальные адреса делятся на *страницы* (*page*)
- Соответствующие единицы в ОП образуют *страничные кадры* (*page frames*)
- В целом, система поддержки страничной ВП памяти называется *пейджингом* (*paging*)
- Передача данных между памятью и диском всегда осуществляется целыми страницами





- Выполняемый процесс обращается по виртуальному адресу $v = (p, s)$
- Механизм отображения ищет номер страницы p в таблице отображения и определяет, что эта страница находится в страничном кадре n, формируя реальный адрес из n и s

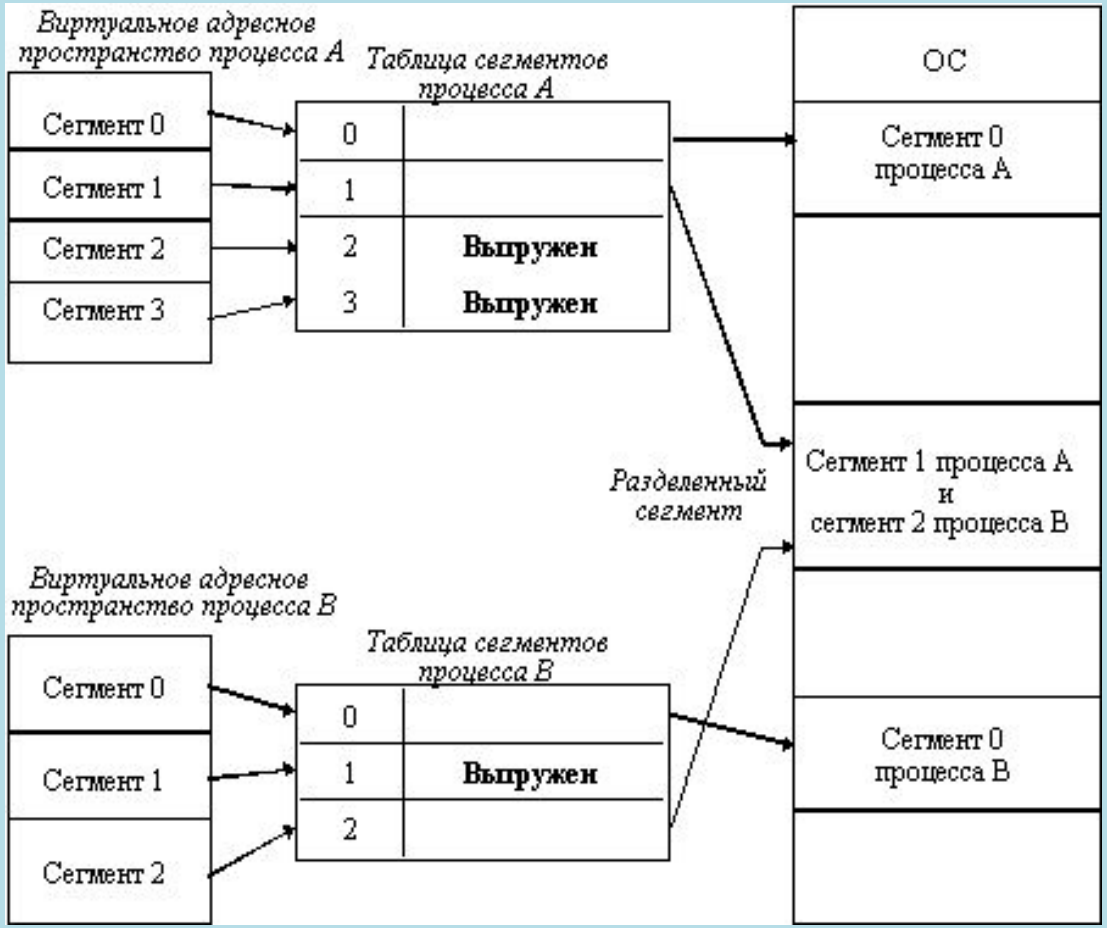
- Отображение должно происходить корректно даже в сложных случаях
- ОС поддерживает одну или несколько (при наличии нескольких сегментов памяти) таблиц страниц для каждого процесса
- При использовании одной таблицы страниц для ссылки на нее обычно используется специальный регистр
- При переключении процессов диспетчер должен найти его таблицу страниц, указатель на которую, таким образом, входит в контекст процесса

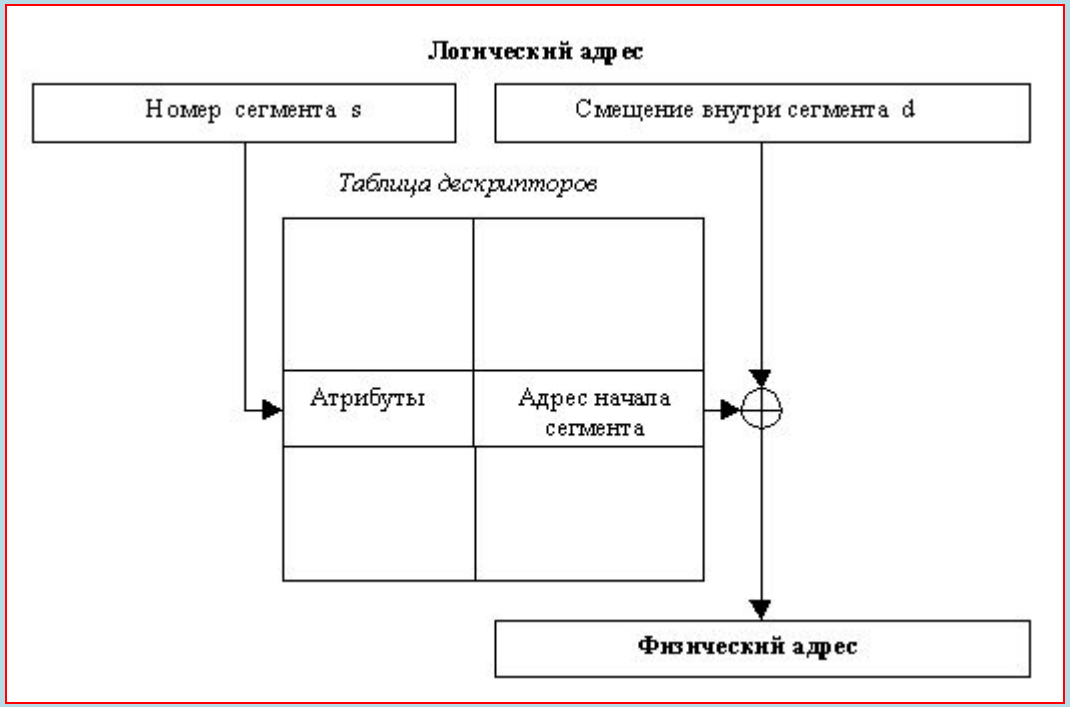
Сегментное распределение памяти

При страничной организации виртуальное адресное пространство процесса делится механически на равные части. Это не позволяет дифференцировать способы доступа к разным частям программы (сегментам).

- С точки зрения пользователя ВП процесса представляется обычно не как линейный массив байтов, а как набор сегментов переменного размера (данные, код, стек)
- Сегментация - схема управления памятью, поддерживающая этот взгляд пользователя
- Сегменты содержат процедуры, массивы, стек или скалярные величины, но обычно не содержат информацию смешанного типа
    - Логическое адресное пространство представляет собой набор сегментов
    - У каждого сегмента имеются имя (номер), размер и другие параметры (уровень привилегий, разрешенные виды обращений, флаги присутствия)
    - Пользователь специфицирует каждый адрес двумя значениями: номером сегмента и смещением
    - Это отличается от схемы пэйджинга, где пользователь задает только один адрес, который разбивается аппаратурой на номер страницы и смещение прозрачным для программиста образом

- Каждому сегменту соответствует линейная последовательность адресов от *0* до максимума
- Различные сегменты могут иметь разные длины, которые могут меняться динамически (например, сегмент стека)
- В элементе таблицы сегментов, помимо физического адреса начала сегмента (если виртуальный сегмент содержится в основной памяти), содержится размер сегмента
- Если размер смещения в виртуальном адресе выходит за пределы размера сегмента, возникает прерывание





- Аппаратная поддержка сегментов относительно слабо распространена (главным образом, на процессорах архитектуры Intel)
- Характеризуется довольно медленной загрузкой селекторов в сегментные регистры, выполняемой при каждом переключении контекста и при каждом переходе между разными сегментами

Странично-сегментное распределение памяти

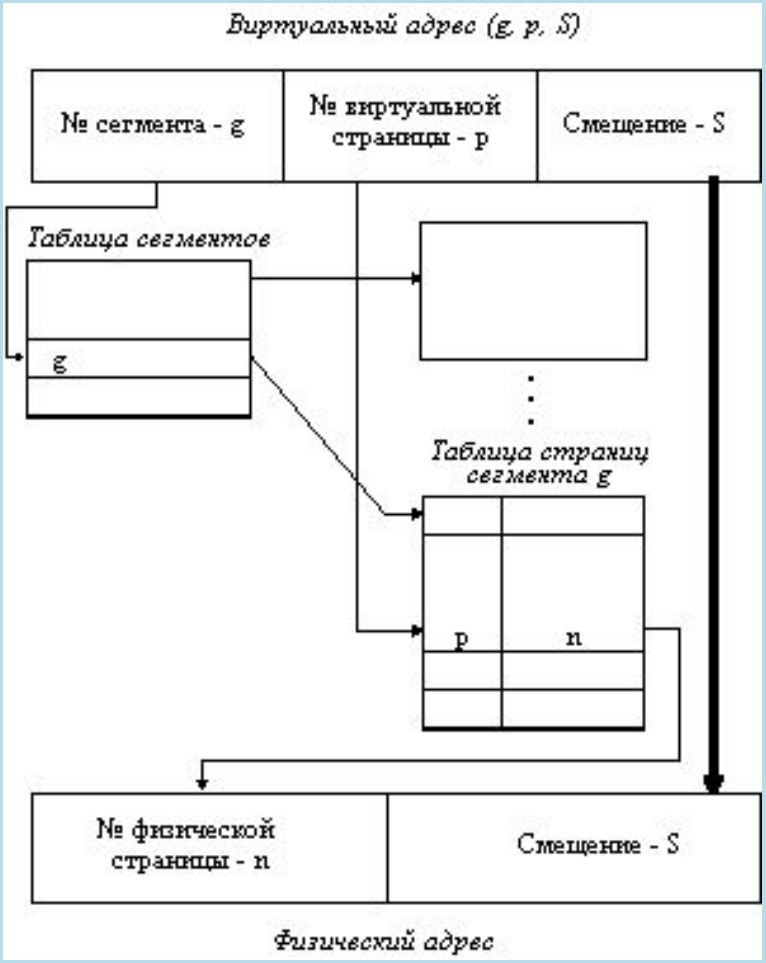
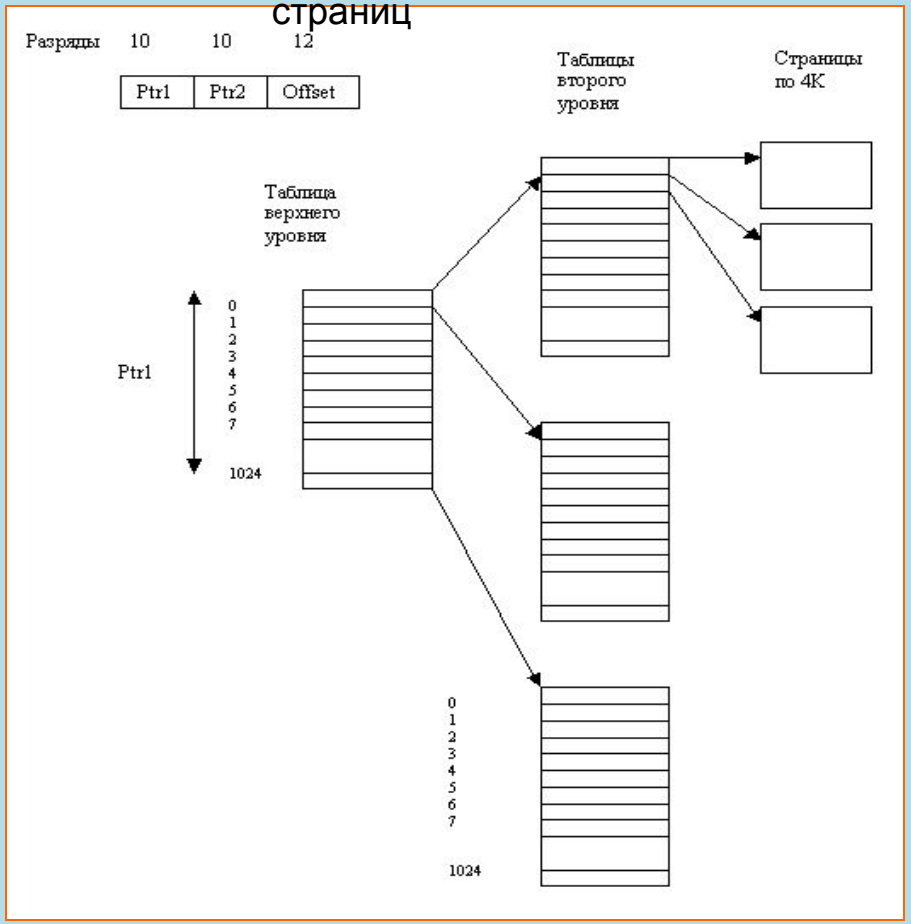


Таблица страниц



Алгоритмы замещения страниц

- В большинстве компьютеров имеются простейшие аппаратные средства, позволяющие собирать некоторую статистику обращений к памяти
- Два специальных флага на каждый элемент таблицы страниц
- *Флаг обращения* (R) автоматически устанавливается, когда происходит любое обращение к странице
- *Флаг изменения* (M) устанавливается, если производится запись в страницу
- Чтобы использовать эти возможности, ОС должна периодически сбрасывать эти флаги

- Существует большое число разнообразных алгоритмов замещения страниц
- Они делятся на ***локальные*** и ***глобальные***
- Локальные алгоритмы распределяют фиксированное или динамически настраиваемое число страниц для каждого процесса
- Когда процессу требуется новая страница в ОП, система удаляет из ОП одну из его страниц, а не из страниц других процессов
- Глобальные алгоритмы замещения выбирают для замещения физическую страницу независимо от того, какому процессу она принадлежит

Алгоритм Opt

Идеальный алгоритм заключается в том, что бы выгружать ту страницу, которая будет запрошена позже всех.

Алгоритм FIFO

- Каждой странице присваивается временная метка
- Реализуется путем создания очереди страниц, в конец которой страницы попадают при загрузке в ОП, а из начала берутся, когда требуется освободить память
- Для замещения выбирается старейшая страница
- Стратегия с достаточной вероятностью может приводить к замещению активно используемых страниц, например, страниц текстового процессора
- При замещении активных страниц все работает корректно, но немедленно происходит *page fault*

- Интуитивно ясно, что чем больше страничных кадров имеет память, тем реже будут иметь место страничные нарушения
- Удивительно, но это не всегда так
- Как установил Биледи, некоторые последовательности обращений к страницам в действительности приводят к увеличению числа страничных нарушений при увеличении кадров, выделенных процессу
- Это явление носит название аномалии *FIFO*

Аномалия Биледи

0 1 2 3 0 1 4 0 1 2 3 4

Самая старая страница	0	1	2	3	0	1	4	4	4	2	3	3
		0	1	2	3	0	1	1	1	4	2	2
Самая новая страница			0	1	2	3	0	0	0	1	4	4

p p p p p p p p p *9 page fault'ов*

(a)

0 1 2 3 0 1 4 0 1 2 3 4

Самая старая страница	0	1	2	3	3	3	4	0	1	2	3	4
		0	1	2	2	2	3	4	0	1	2	3
			0	1	1	1	2	3	4	0	1	2
Самая новая страница				0	0	0	1	2	3	4	0	1

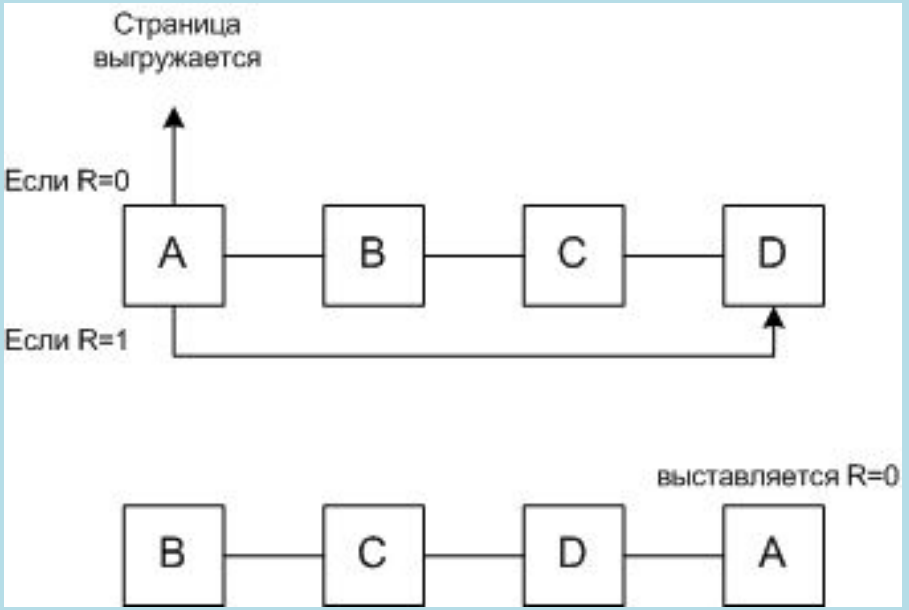
p p p p p p p p p p *10 page fault'ов*

(b)

- При выборе стратегии *FIFO* для строки обращений к памяти 012301401234 три кадра (9 faults) оказываются лучше чем 4 кадра (10 faults)
- Аномалию *FIFO* следует считать курьезом, иллюстрирующим сложность ОС, где интуитивный подход не всегда приемлем

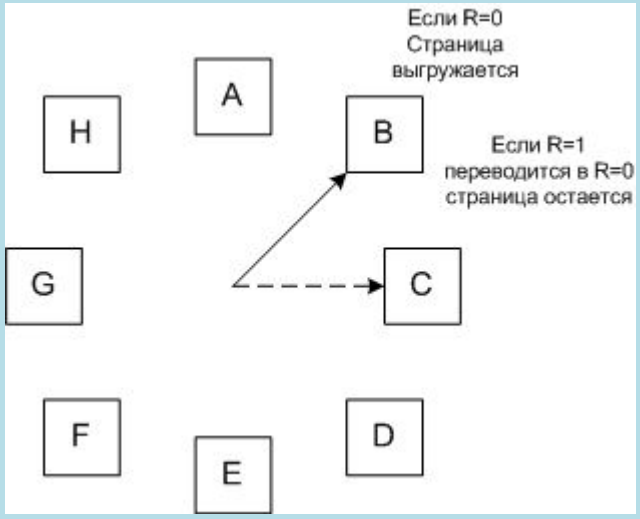
Алгоритм "вторая попытка"

Подобен FIFO, но если R=1, то страница переводится в конец очереди, если R=0, то страница выгружается.
В этом случае активно используемая страница не покинет ОП, но каждый раз происходит перемещение страницы.



Алгоритм “часы"

Позволяет не перестраивать очередь.
Активные страницы остаются в памяти.



Алгоритм LRU

- Нужно иметь связанный список всех страниц в памяти, в начале которого будут часто используемые страницы
- Список должен обновляться при каждой обращении;
- 64-битный регистр, значение которого увеличивается на 1 после выполнения каждой инструкции
- В таблице страниц - поле, в которое заносится значение регистра при каждом обращении к странице
- При возникновении *page fault* выгружается страница с наименьшим значением этого поля

Алгоритм LFU

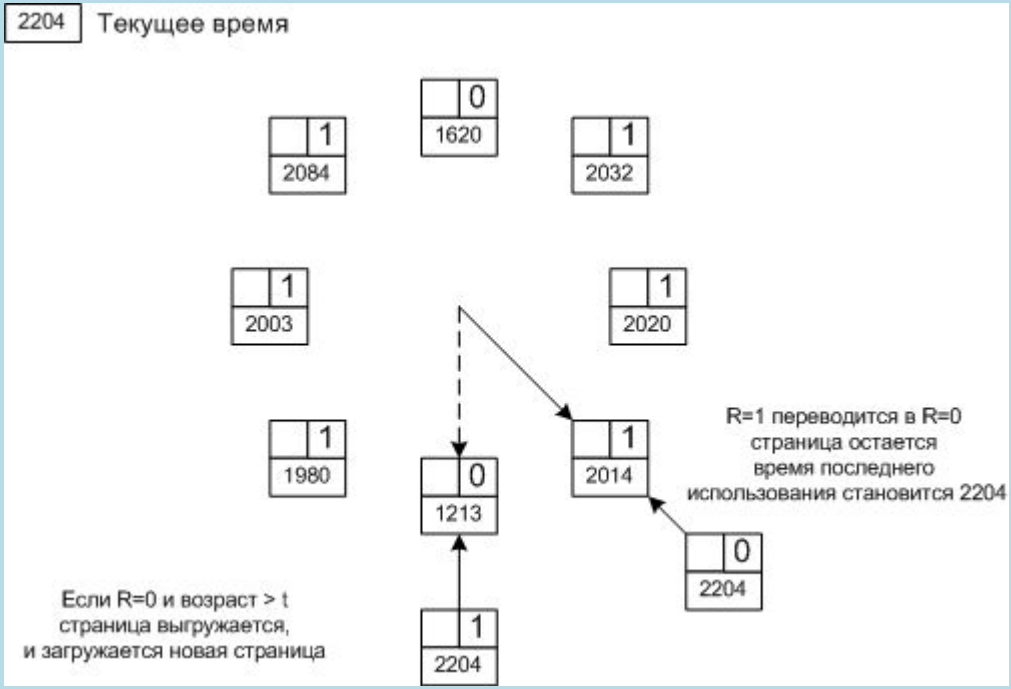
- Выталкивается та страница, которая наименее интенсивно используется или обращения к которой наименее часты.
- Подобный подход опять-таки кажется интуитивно оправданным, однако в то же время велика вероятность того, что удаляемая страница будет выбрана нерационально. Например, наименее интенсивно используемой может оказаться та страница, которую только что переписали в основную память и к которой успели обратиться только один раз, в то время как к другим страницам могли уже обращаться более одного раза.

Алгоритм NUR

Основан на алгоритме «Часы».

$$R = \begin{cases} 0, \text{ если к странице не было обращений} \\ 1, \text{ если к странице были обращения} \end{cases}$$

$$M = \begin{cases} 0, \text{ если страница не изменялась} \\ 1, \text{ если страница изменялась} \end{cases}$$



Группа 1	обращений не было	модификаций не было
Группа 2	обращений не было	модификация была
Группа 3	обращения были	модификаций не было
Группа 4	обращения были	модификация была