Раздел 3

Управление памятью

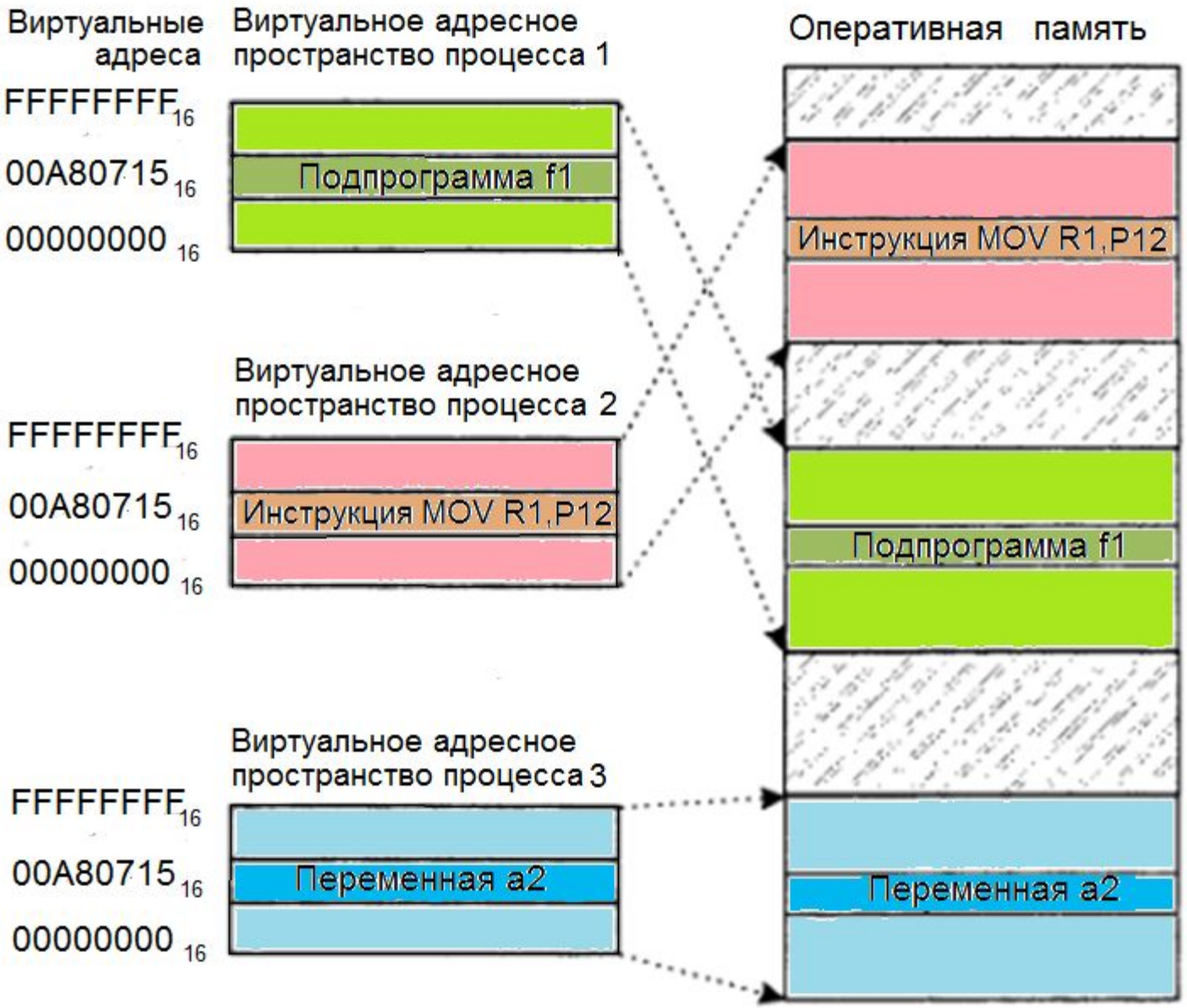
Функции ОС по управлению памятью

- отслеживание свободной и занятой памяти
-выделение памяти процессам и освобождение памяти по завершении процессов
- организация виртуальной памяти
- настройка адресов программы на конкретную область физической памяти
- динамическое распределение памяти
- дефрагментация памяти
- защита памяти

На разных этапах жизненного цикла программы используются различные типы адресов:

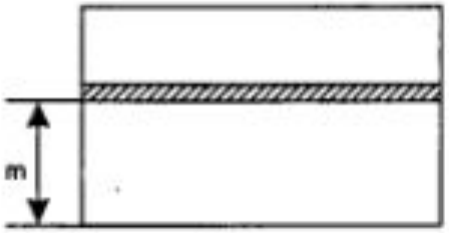
- символьные имена присваивает пользователь при написании программы на алгоритмическом языке или ассемблере
- виртуальные адреса
- вырабатывает транслятор, переводящий программу на машинный язык
- физические адреса соответствуют номерам ячеек оперативной памяти, где в действительности расположены или будут расположены переменные и команды

Совокупность виртуальных адресов процесса называется *виртуальным адресным пространством*.

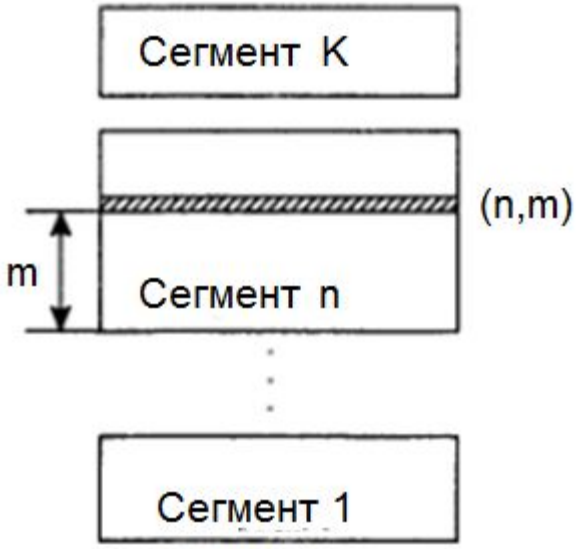

Виртуальные адреса
Виртуальное адресное пространство процесса 1
Оперативная память
FFFFFFFF16
00A80715 16
00000000 16
Подпрограмма f1
Инструкция MOV R1,P12
Виртуальное адресное пространство процесса 2
FFFFFFFF16
00A80715 16
00000000 16
Инструкция MOV R1,P12
Подпрограмма f1
Виртуальное адресное пространство процесса 3
FFFFFFFF16
00A80715 16
00000000 16
Переменная a2
Переменная a2

Структуризация виртуального адресного пространства

- Плоское (flat) виртуальное адресное пространство (ВАП) *Виртуальный адрес – смещение от начала ВАП,* одно число.

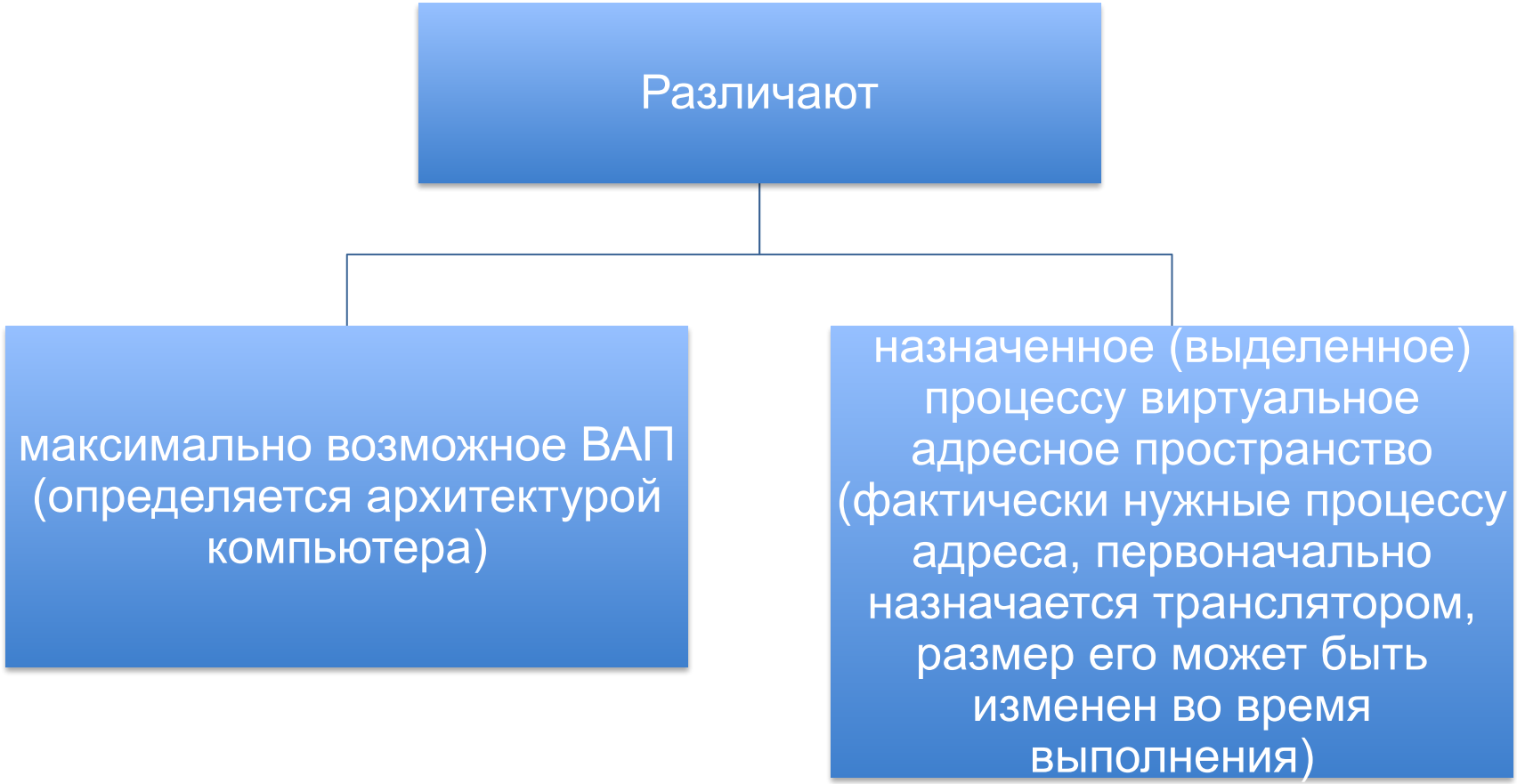


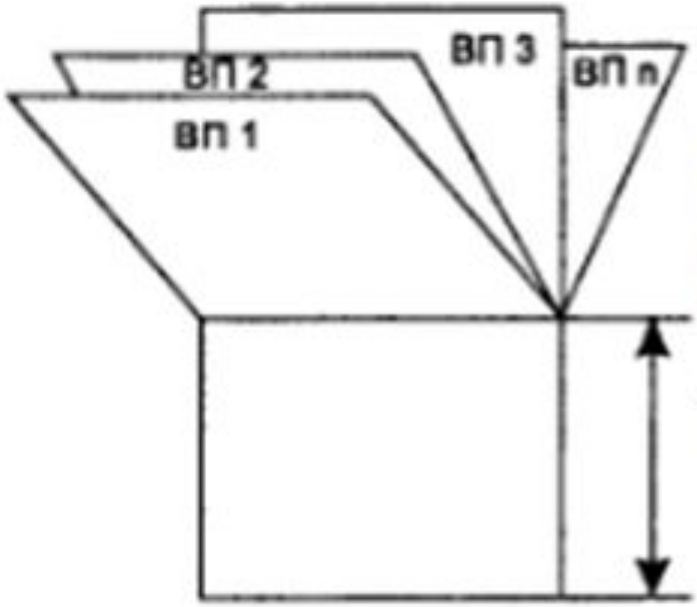
- Сег[illegible] ВАП



Виртуальный адрес – пара чисел (n,m), где n определяет сегмент, а m – смещение внутри сегмента.

Задачей ОС является отображение индивидуальных виртуальных адресных пространств всех одновременно выполняющихся процессов на общую физическую память


Различают
максимально возможное ВАП (определяется архитектурой компьютера)
назначенное (выделенное) процессу виртуальное адресное пространство (фактически нужные процессу адреса, первоначально назначается транслятором, размер его может быть изменен во время выполнения)

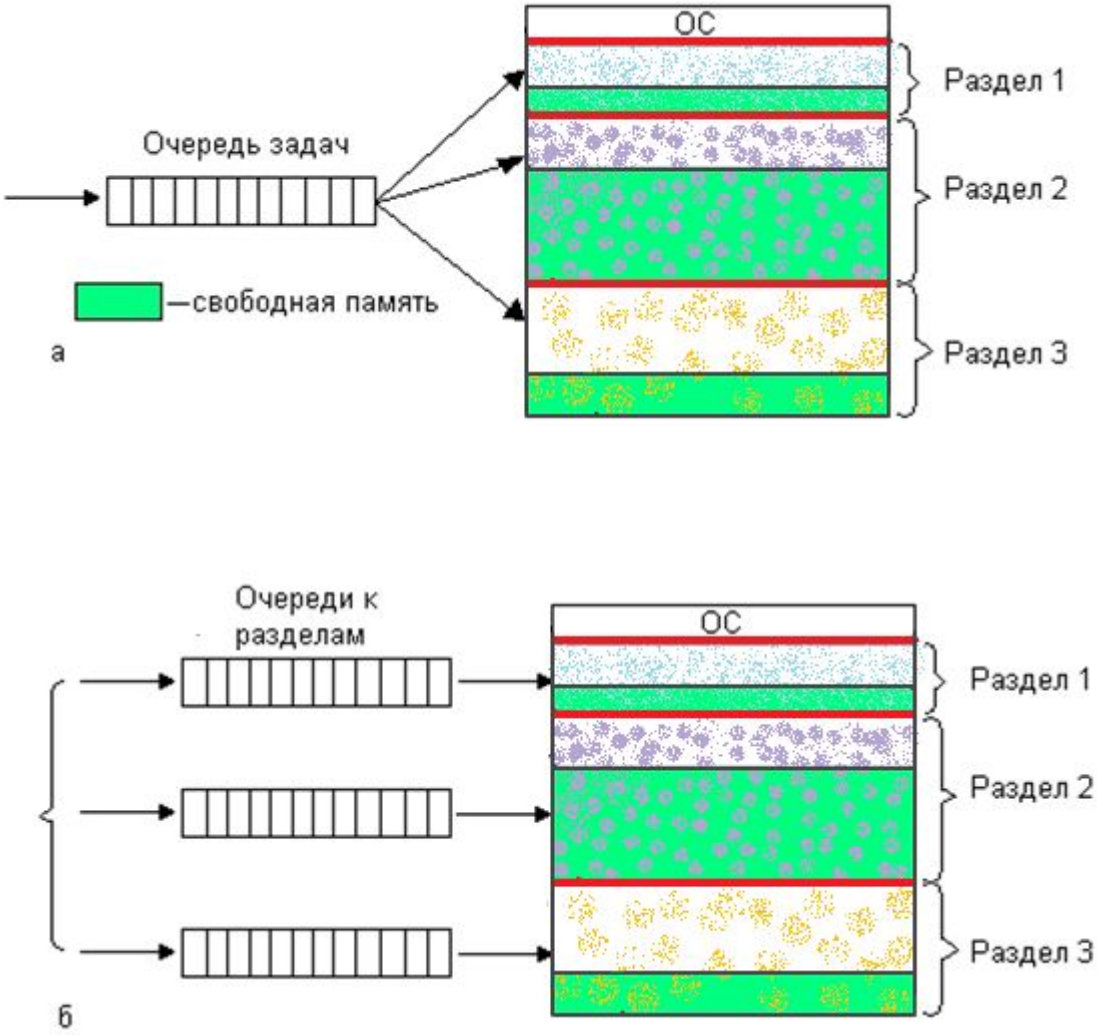


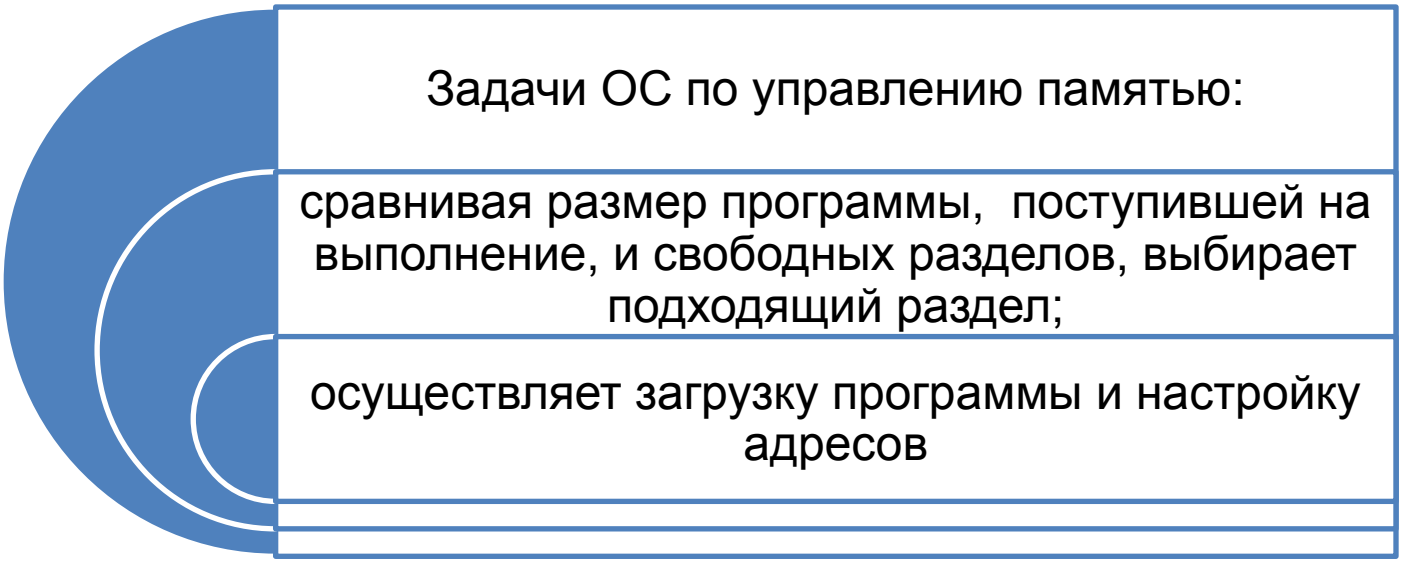
Индивидуальные
части ВАП
(пользовательские, по 2 Гбайт)
00000000-7FFFFFFF

Общая (системная)
часть ВАП
80000000-FFFFFFFF

Алгоритмы распределения памяти

Распределение памяти фиксированными разделами





•+

•Достоинства:

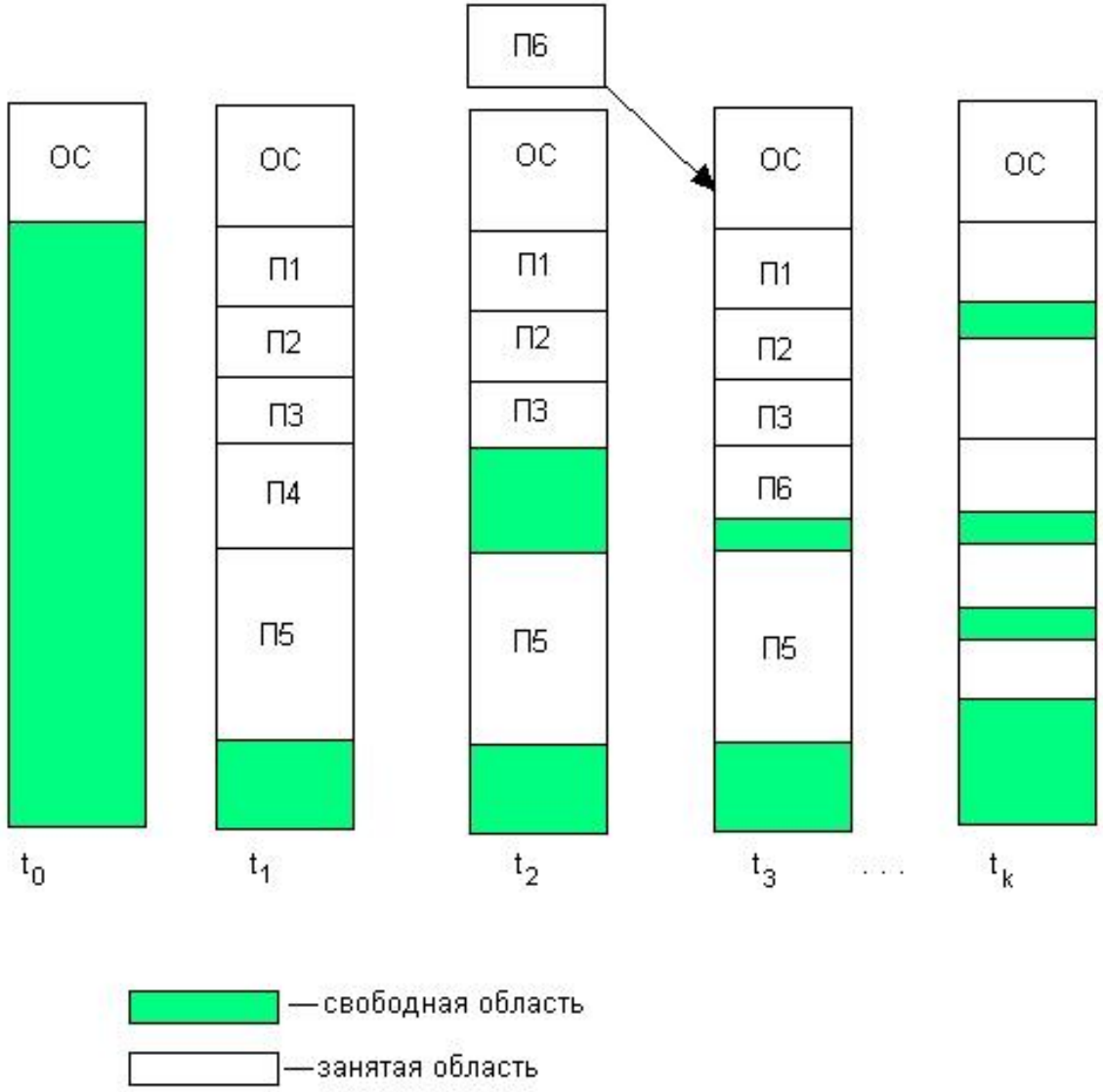
• простота реализации

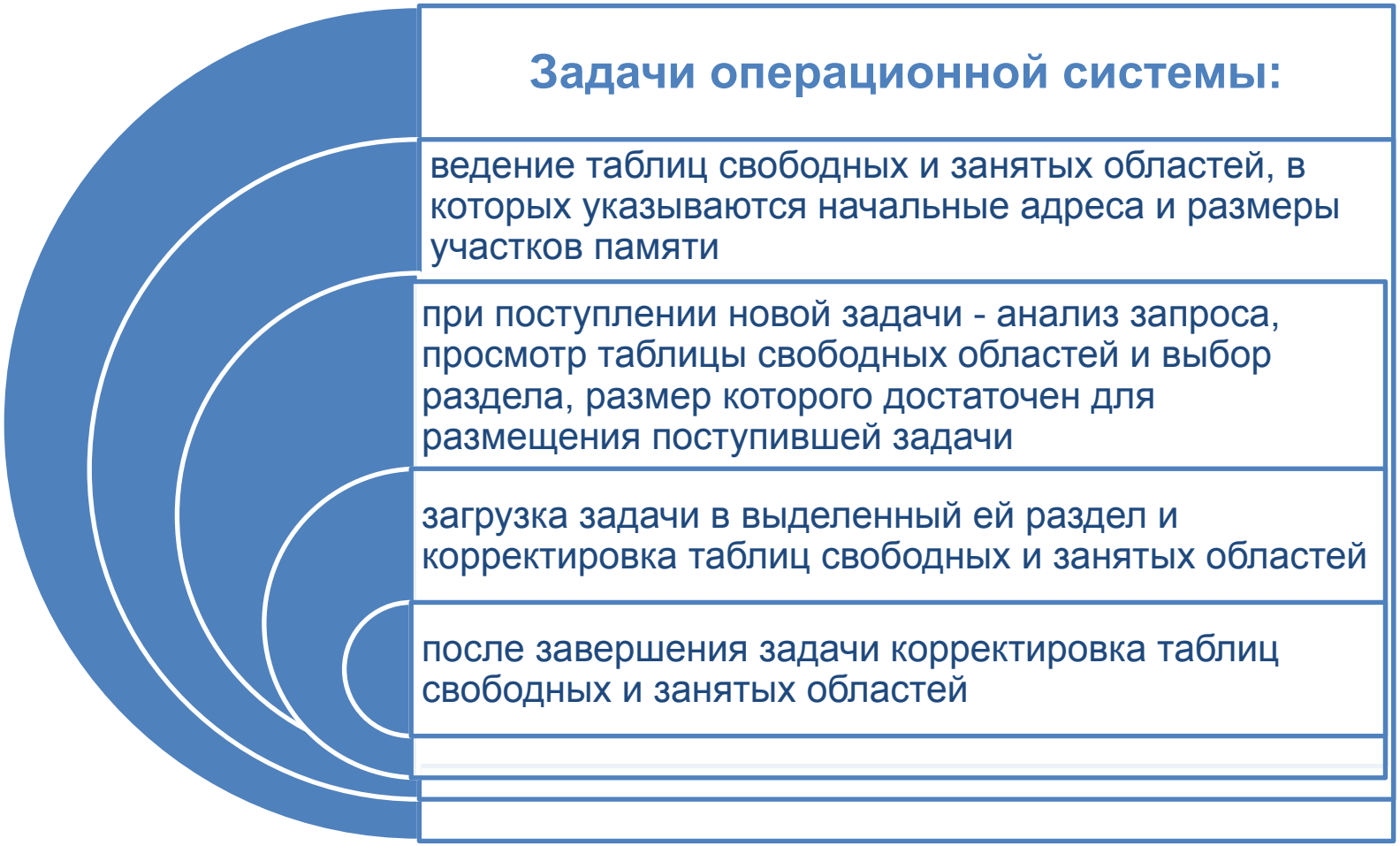
•-

•Недостатки:

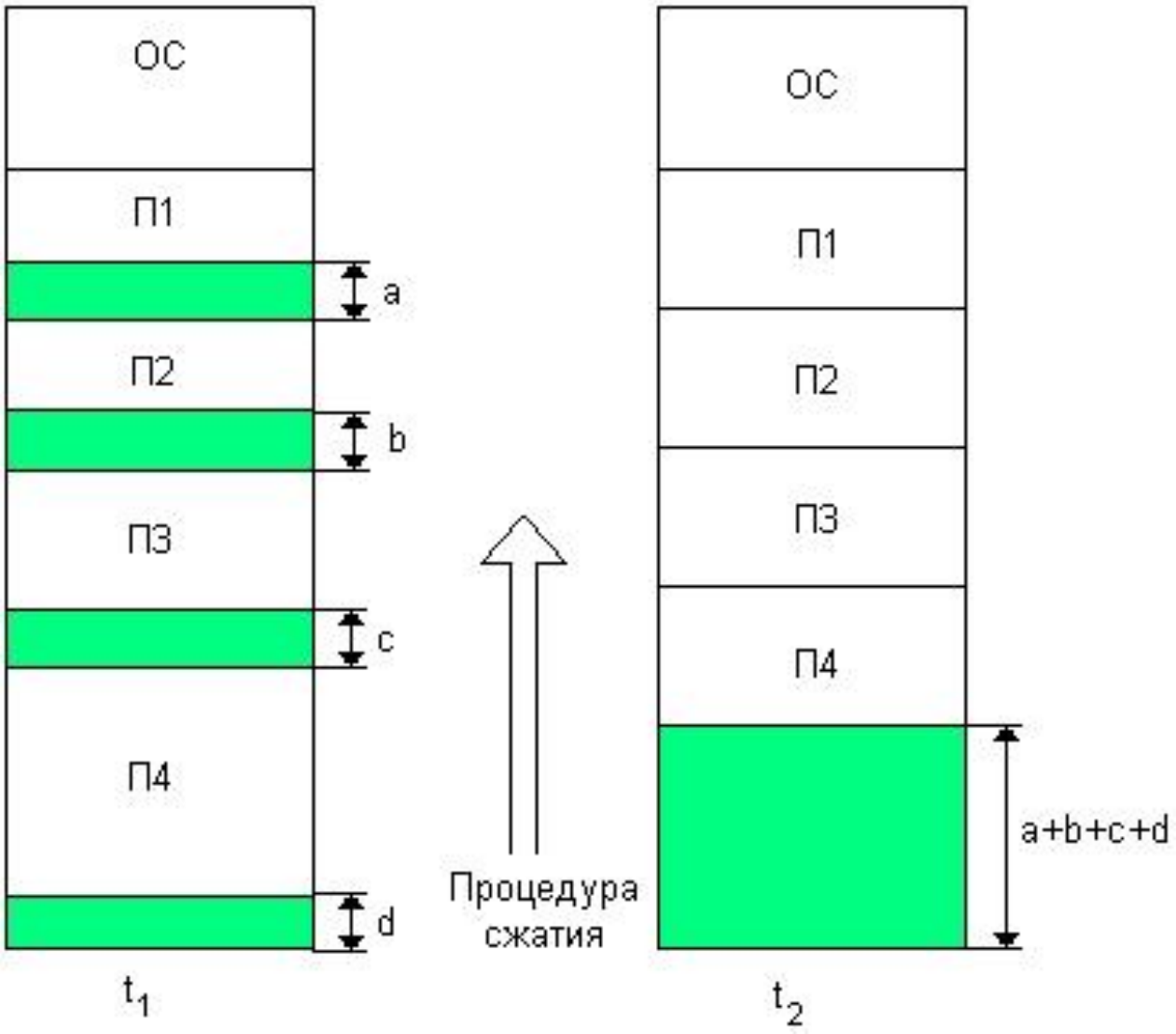
• -жесткость. В каждом разделе может выполняться только одна программа.

Распределение памяти динамическими разделами




Задачи операционной системы:
ведение таблиц свободных и занятых областей, в которых указываются начальные адреса и размеры участков памяти
при поступлении новой задачи - анализ запроса, просмотр таблицы свободных областей и выбор раздела, размер которого достаточен для размещения поступившей задачи
загрузка задачи в выделенный ей раздел и корректировка таблиц свободных и занятых областей
после завершения задачи корректировка таблиц свободных и занятых областей

Распределение памяти перемещаемыми разделами



Виртуализация оперативной памяти осуществляется совместно ОС и аппаратными средствами процессора и включает решение следующих задач:

1. • размещение данных в запоминающих устройствах разного типа, например часть кодов программы – в оперативной памяти, а часть – на диске;
2. • выбор образов процессов или их частей для перемещения из оперативной памяти на диск и обратно;
3. • перемещение по мере необходимости данных между памятью и диском;
4. • преобразование виртуальных адресов в физические.

Виртуализация памяти может быть осуществлена на основе двух различных подходов:

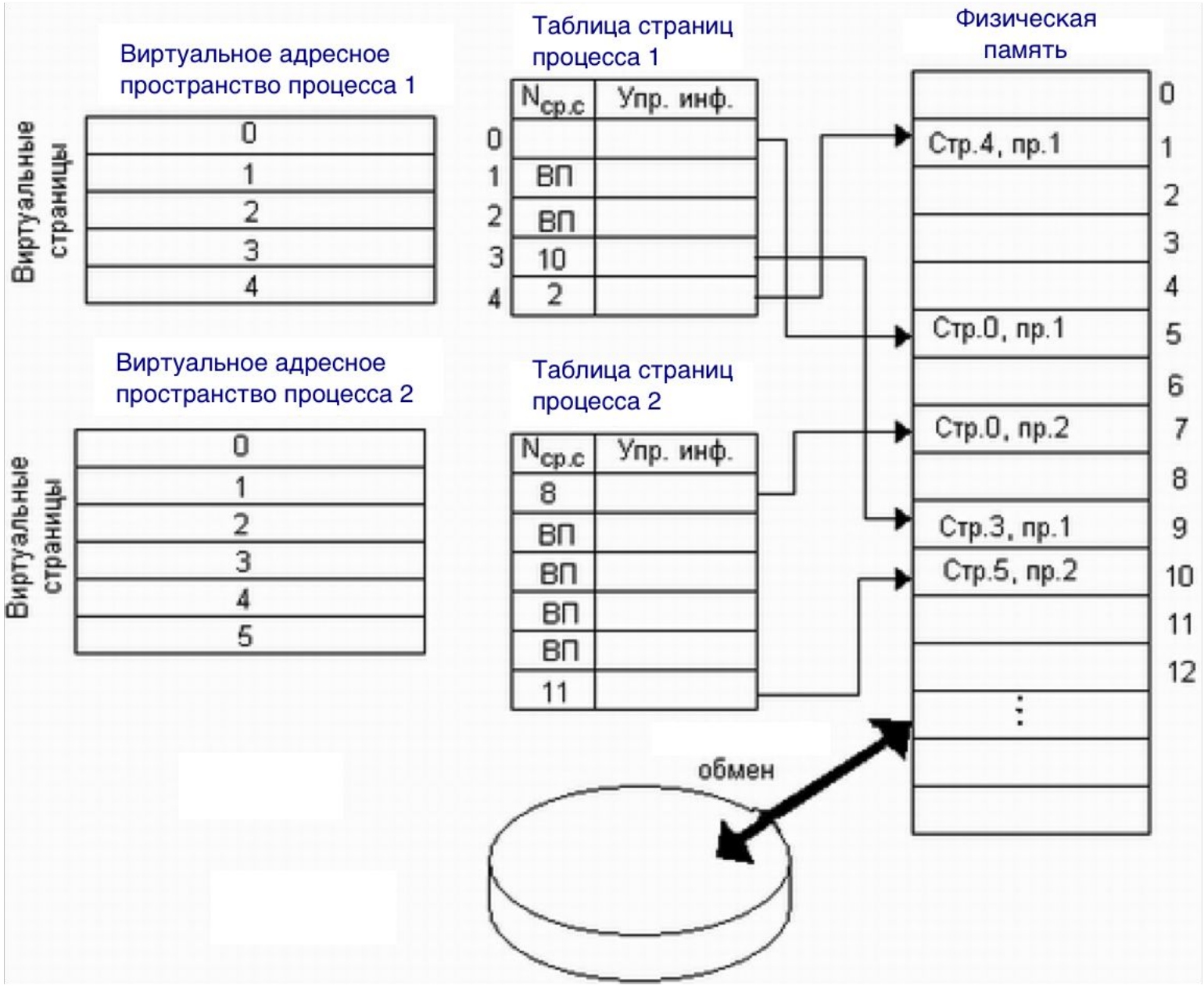
свопинг – образы процессов выгружаются на диск и возвращаются в оперативную память целиком

виртуальная память – между оперативной памятью и диском перемещаются части (сегменты, страницы и т.п.) образов процессов

Страничное распределение

Виртуальное адресное пространство процесса делится на части одинакового, фиксированного для данной системы размера, называемые виртуальными страницами.


Дескриптор страницы
номер физической страницы, в которую загружена данная виртуальная страница
признак присутствия
признак модификации
признак обращения


Виртуальное адресное пространство процесса 1
Виртуальные страницы
0
1
2
3
4
Виртуальное адресное пространство процесса 2
Виртуальные страницы
0
1
2
3
4
5
Таблица страниц процесса 1
Nср.с
Упр. инф.
0
1 ВП
2 ВП
3 10
4 2
Таблица страниц процесса 2
Nср.с
Упр. инф.
8
ВП
ВП
ВП
ВП
11
Физическая память
0
Стр.4, пр.1
1
2
3
4
Стр.0, пр.1
5
6
Стр.0, пр.2
7
8
Стр.3, пр.1
9
Стр.5, пр.2
10
11
12
обмен

Механизм преобразования виртуального адреса в физический при страничной организации памяти

При каждом обращении к оперативной памяти аппаратными средствами выполняются следующие действия:

на основании начального адреса таблицы страниц (содержимое регистра адреса таблицы страниц), номера виртуальной страницы (старшие разряды виртуального адреса) и длины записи в таблице страниц (системная константа) определяется адрес нужной записи в таблице

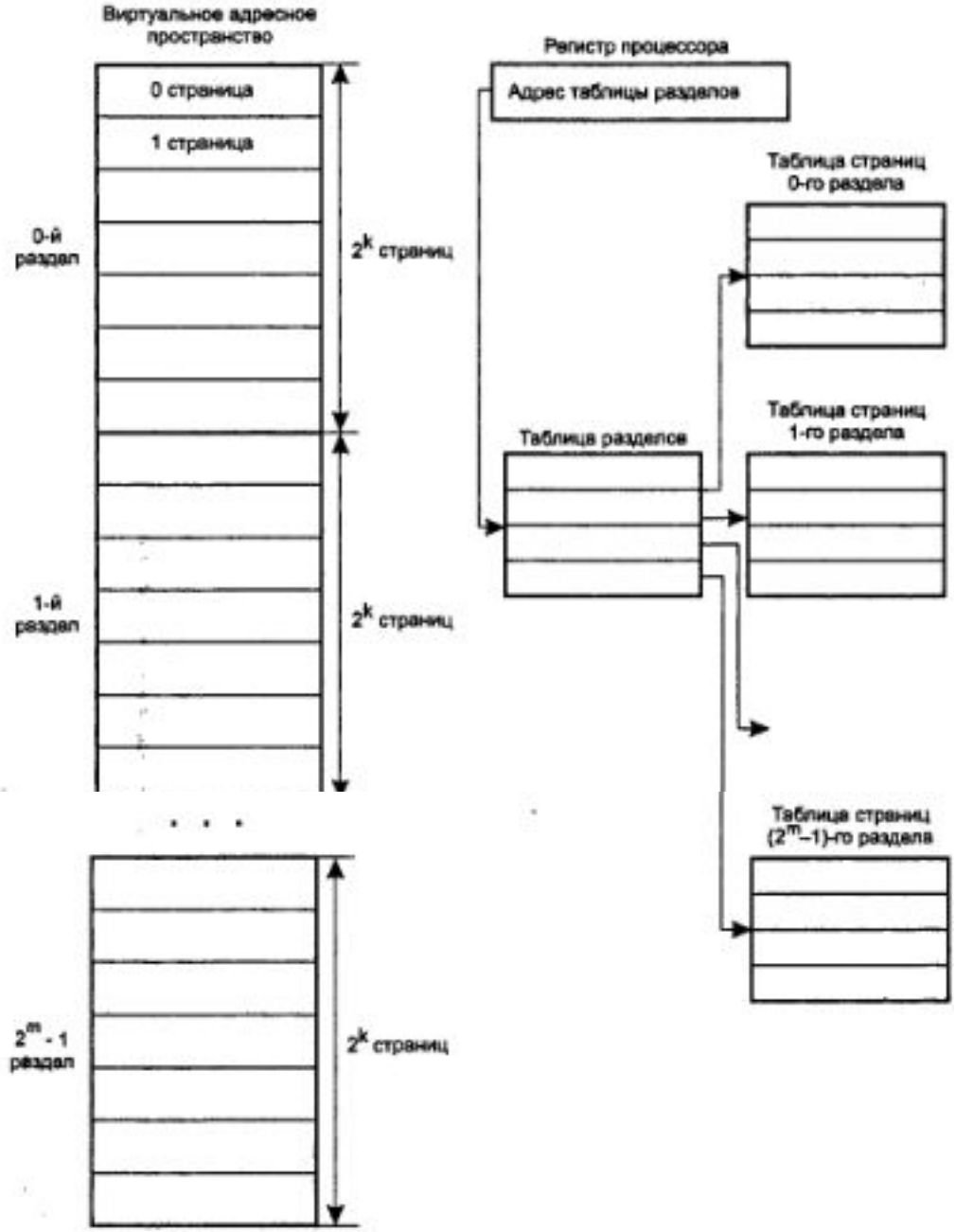
из этой записи извлекается номер физической страницы

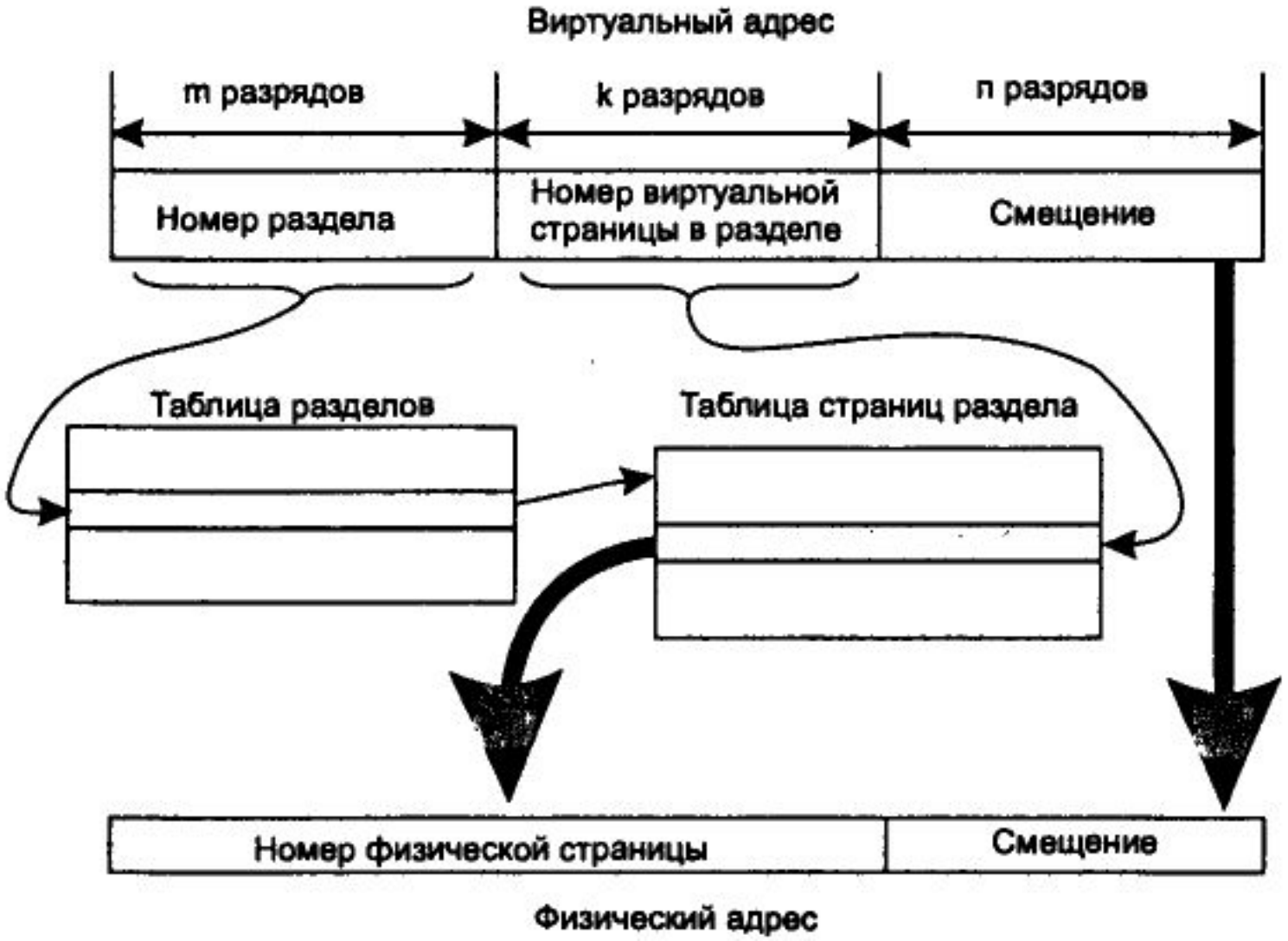
к номеру физической страницы присоединяется смещение (младшие разряды виртуального адреса)

Таблицы страниц для больших объемов памяти

При размере страниц в 4 Кбайт, 32-разрядное адресное пространство имеет 1 миллион страниц, а 64-разрядное адресное пространство состоит из 2^{52}страниц. Таблица страниц должна содержать запись о каждой виртуальной странице. Если каждая запись будет занимать 8 байт, то размер таблицы превысит 8 миллионов байт для 32 разрядной архитектуры и 30 миллионов байт для 64-разрядной. При этом каждому процессу требуется своя собственная таблица страниц.

- Многоуровневые таблицы страниц


Виртуальное адресное пространство
0 страница
1 страница
0-й раздел
2^k страниц
1-й раздел
2^k страниц
$2^m - 1$ раздел
2^k страниц
Регистр процессора
Адрес таблицы разделов
Таблица разделов
Таблица страниц 0-го раздела
Таблица страниц 1-го раздела
Таблица страниц (2^m-1)-го раздела


Виртуальный адрес
m разрядов
k разрядов
n разрядов
Номер раздела
Номер виртуальной страницы в разделе
Смещение
Таблица разделов
Таблица страниц раздела
Номер физической страницы
Смещение
Физический адрес

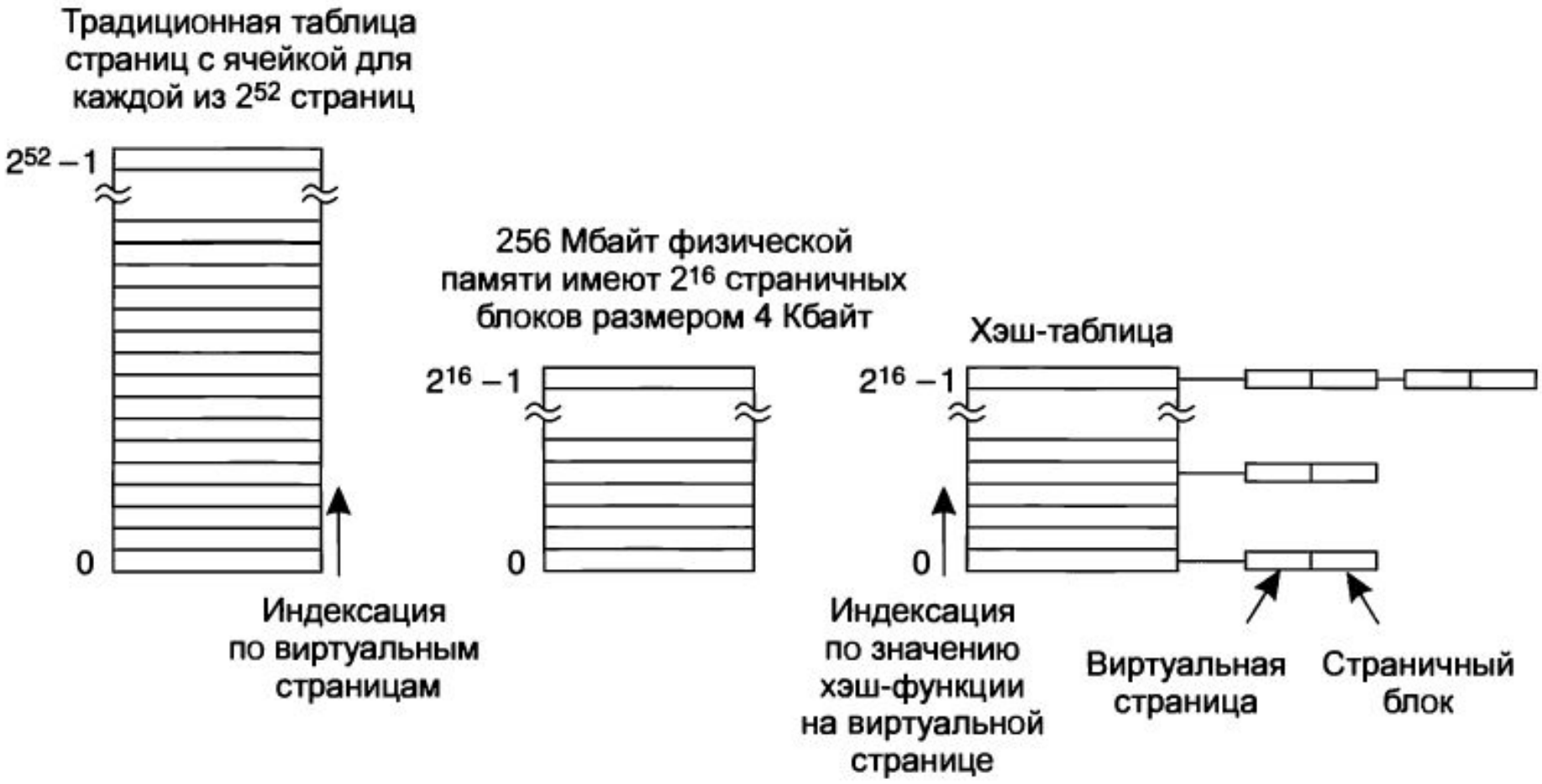
Буфер быстрого преобразования адреса TLB

Кэширование пар номеров виртуальных и физических страниц в буфере ассоциативной трансляции TLB (Translation Lookaside Buffer) **позволяет ускорять преобразование виртуальных адресов в физические.**

TLB представляет собой ассоциативную память небольшого объема, предназначенную для хранения интенсивно используемых дескрипторов страниц.

В буфере TLB кэшируются дескрипторы страниц из таблицы страниц
Для хранения дескриптора в кэше отводится одна *строка*. Каждая строка дополнена *тегом*, в котором содержится номер соответствующей виртуальной страницы.

Инвертированные таблицы страниц



Алгоритмы замещения страниц

1. Оптимальный алгоритм замещения страниц

Каждая страница должна быть помечена количеством команд, которые выполняются до первого обращения к странице.

Суть алгоритма:

на выгрузку выбирается страница, имеющая пометку с наибольшим значением.

2. Алгоритм исключения недавно использовавшейся страницы NRU

Управляющая информация в дескрипторе страницы:
Бит R (обращения), бит M (модификации).

Суть алгоритма

1. При запуске процесса R=0, M=0 для всех его страниц.
2. При каждом прерывании по таймеру бит R сбрасывается, чтобы отличить те страницы, к которым в последнее время не было обращений, от тех, к которым такие обращения были.
3. При возникновении ошибки отсутствия страницы ОС просматривает все дескрипторы страниц и делит их на четыре категории:

Класс 0: в последнее время не было ни обращений, ни модификаций

Класс 1: обращений в последнее время не было, но страница модифицирована.

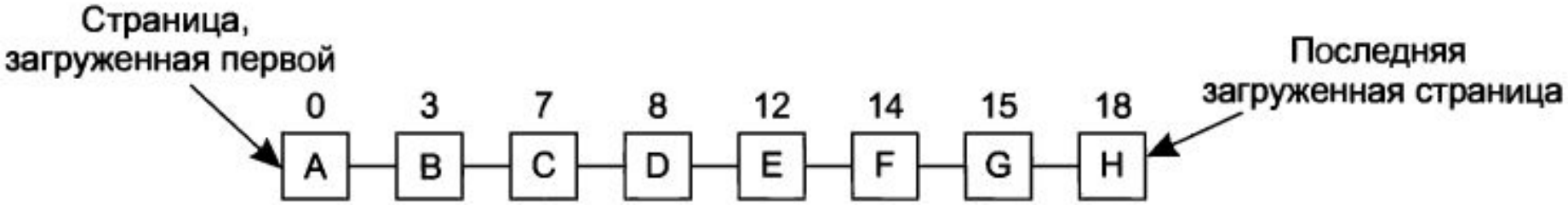
Класс 2: в последнее время были обращения, но модификаций не было.

Класс 3: в последнее время были и обращения и модификации.

Алгоритм исключения недавно использовавшейся страницы — NRU(Not Recently Used) удаляет произвольную страницу, относящуюся к самому низкому непустому классу.

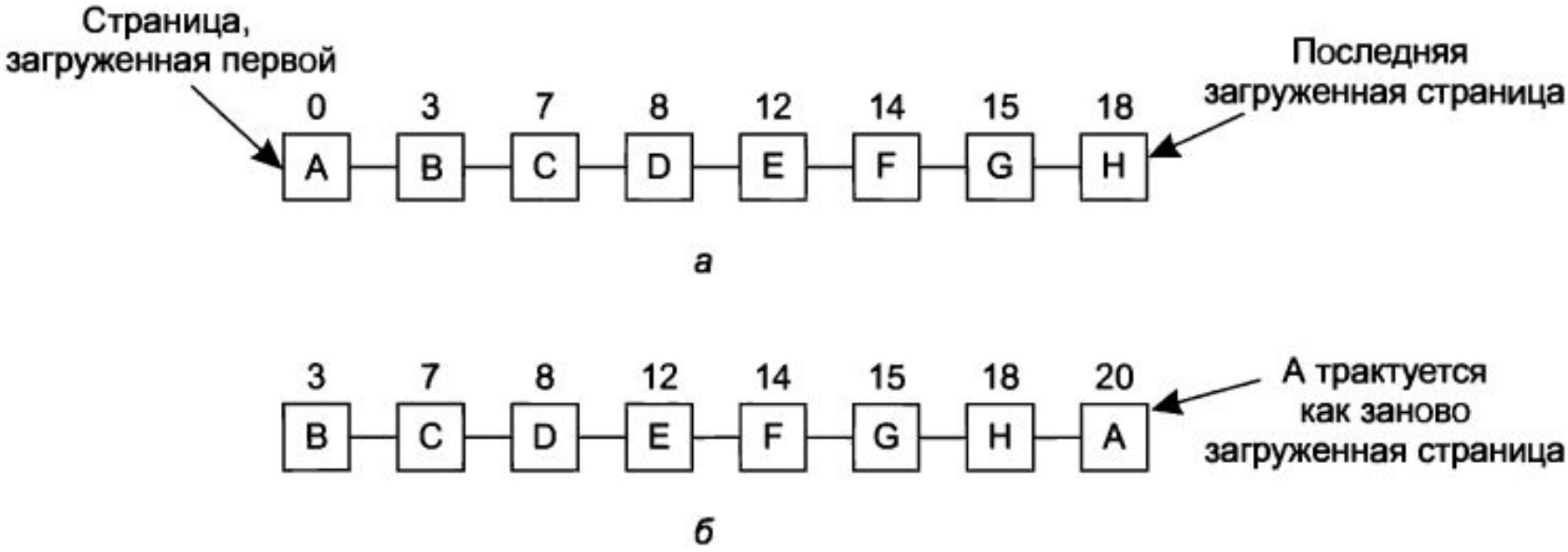
3. Алгоритм «первой пришла, первой и ушла» FIFO

Суть алгоритма



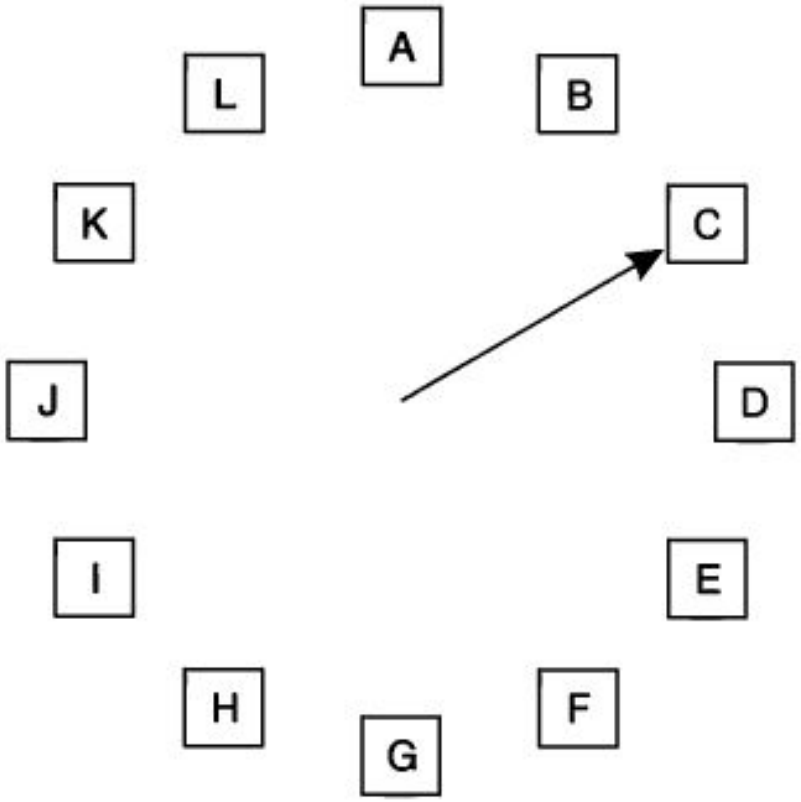
ОС ведет список всех страниц, находящихся на данный момент в памяти, причем совсем недавно поступившие находятся в хвосте, поступившие раньше всех — в голове списка. При возникновении ошибки отсутствия страницы удаляется страница, находящаяся в голове списка, а к его хвосту добавляется новая страница. Принцип FIFO в чистом виде используется довольно редко.

4. Алгоритм «второй шанс»



Простая модификация алгоритма FIFO, исключающая проблему удаления часто востребуемой страницы - проверка бита R самой старой страницы. Если R=0, значит, страница не только старая, но и невостребованная, поэтому она тут же удаляется. Если бит R =1, он сбрасывается, а страница помещается в конец списка страниц, и время ее загрузки обновляется, как будто она только что поступила в память. Затем поиск продолжается.

5. Алгоритм «часы»



Когда происходит страничное прерывание, поверяется страница, на которую указывает стрелка. Предпринимаемые действия зависят от бита R:

R=0: страница выгружается

R=1: бит R сбрасывается, стрелка движется вперед

6. Алгоритм замещения наименее востребованной страницы LRU (Least Recently Used)

1. Для полной реализации алгоритма необходимо вести связанный список всех страниц, находящихся в памяти. В начале этого списка должна быть только что востребованная страница, а в конце — наименее востребованная. Этот список должен обновляться при каждом обращении к памяти. Для поиска страницы в списке, ее удаления из него и последующего перемещения этой страницы вперед потребуется довольно много времени, даже если это будет возложено на аппаратное обеспечение.

2. В состав аппаратного обеспечения вводим 64-разрядный счетчик С, значение которого автоматически увеличивается после каждой команды. Каждая запись в таблице страниц имеет достаточно большое поле, чтобы содержать значение этого счетчика. После каждого обращения к памяти текущее значение счетчика С сохраняется в дескрипторе страницы, к которой было это обращение. При возникновении ошибки отсутствия страницы ОС ищет наименьшее значение счетчика в таблице страниц. Та страница, к чьей записи относится это значение, и будет наименее востребованной.

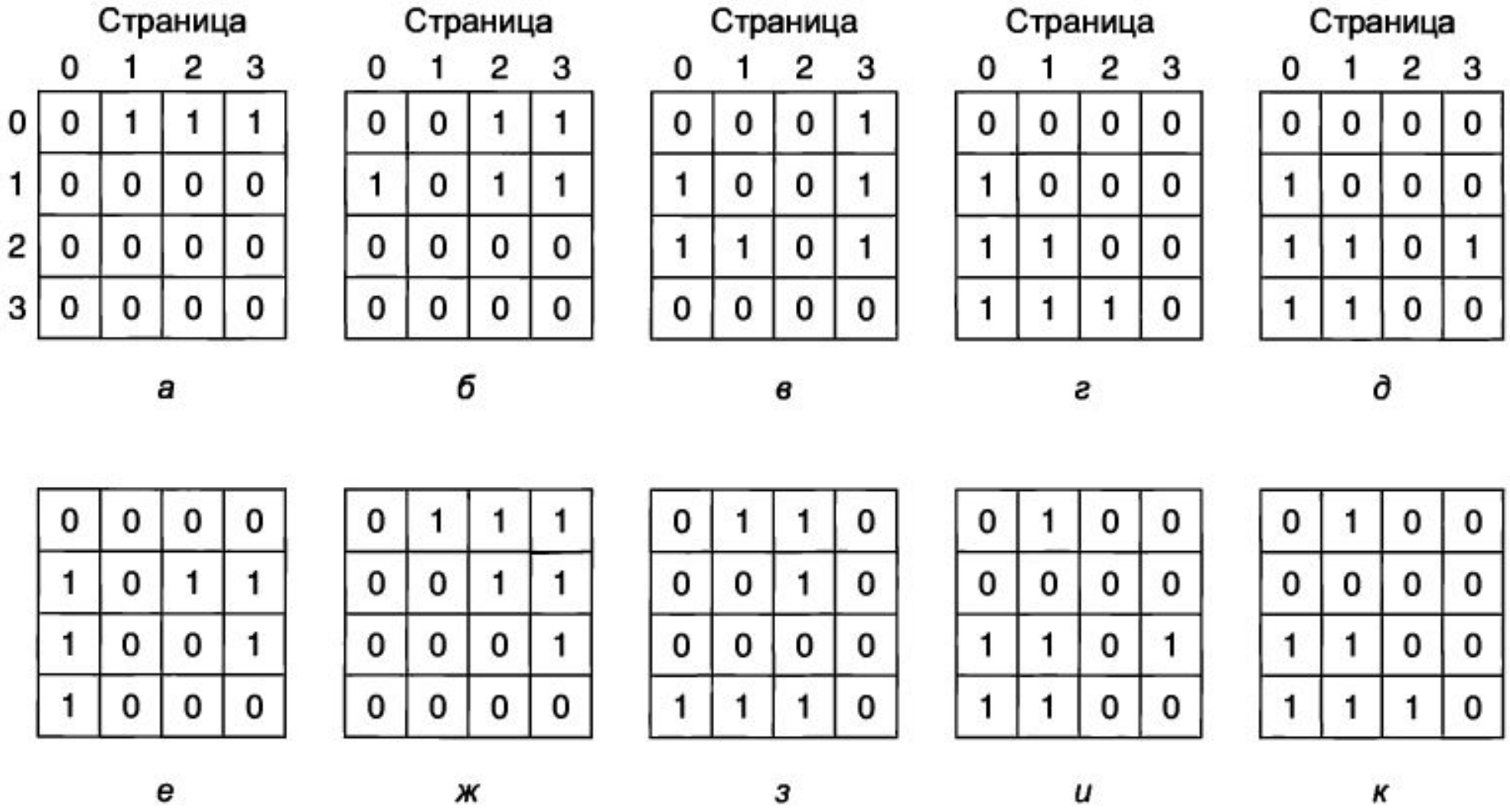
3. В системе всего n штук страничных блоков.

Необходимо аппаратно поддерживать матрицу n X n битов. При обращении к страничному блоку k, аппаратно сначала устанавливаются все биты строки k в 1,а затем обнуляются все биты столбца k.

В любой момент времени строка с наименьшим двоичным значением относится к наименее востребованной странице, строка со следующим наименьшим значением относится к следующей наименее востребованной странице, и т. д

4 страничных блока

Обращение происходит в следующем порядке: 0123210323



4. Моделирование LRU в программном обеспечении

Алгоритм нечастого востребования — NFU (Not Frequently Used)

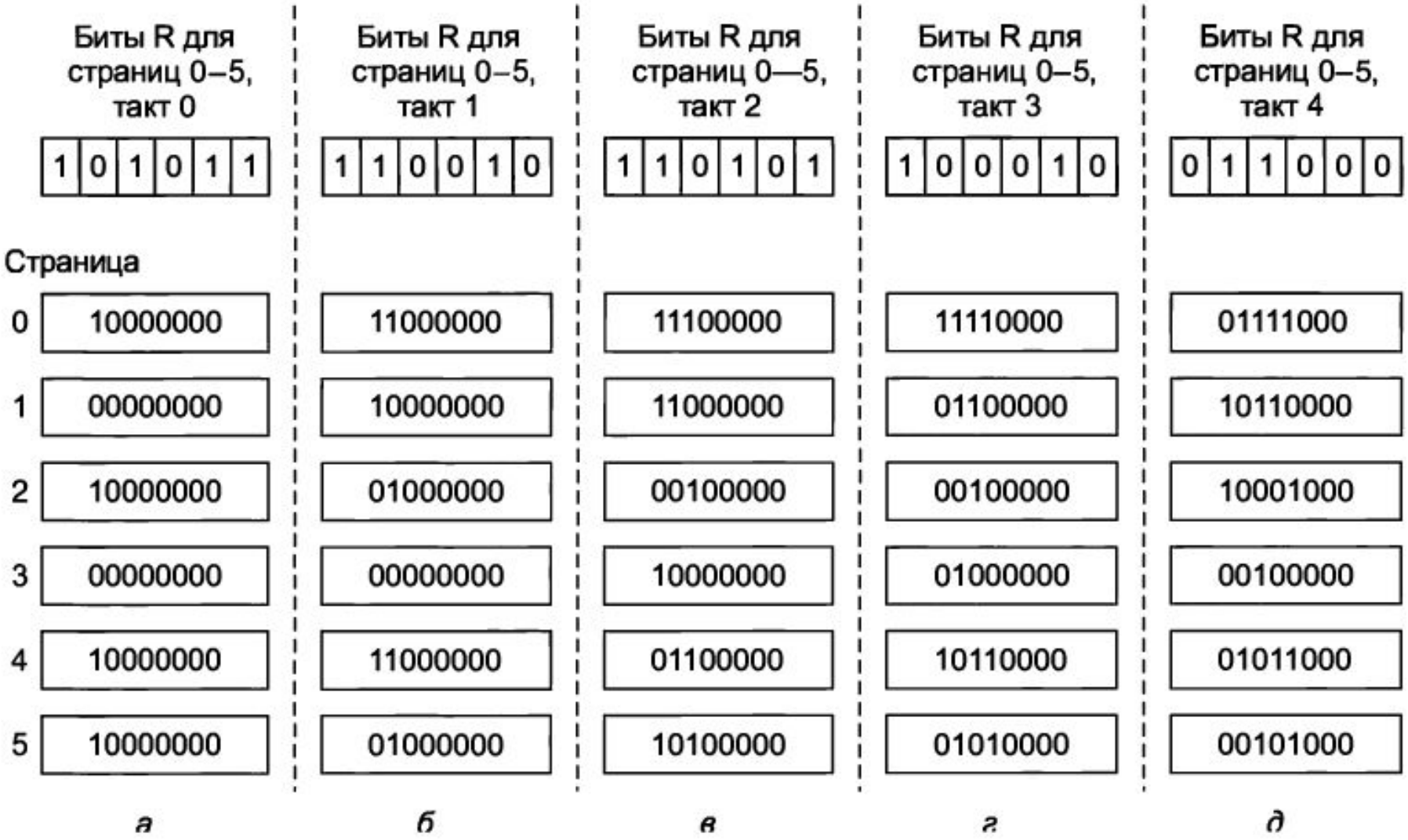
Организуем программный счетчик для каждой страницы. При каждом прерывании от таймера ОС сканирует дескрипторы всех находящихся в памяти страниц. Для каждой страницы к счетчику добавляется значение бита R. Счетчики позволяют приблизительно отследить частоту обращений к каждой странице. При возникновении ошибки отсутствия страницы для замещения выбирается та страница, чей счетчик имеет наименьшее значение.

он никогда ничего не забывает.

Модифицированный алгоритм NFU (алгоритм старения) позволяет достаточно близко подойти к имитации алгоритма LRU.

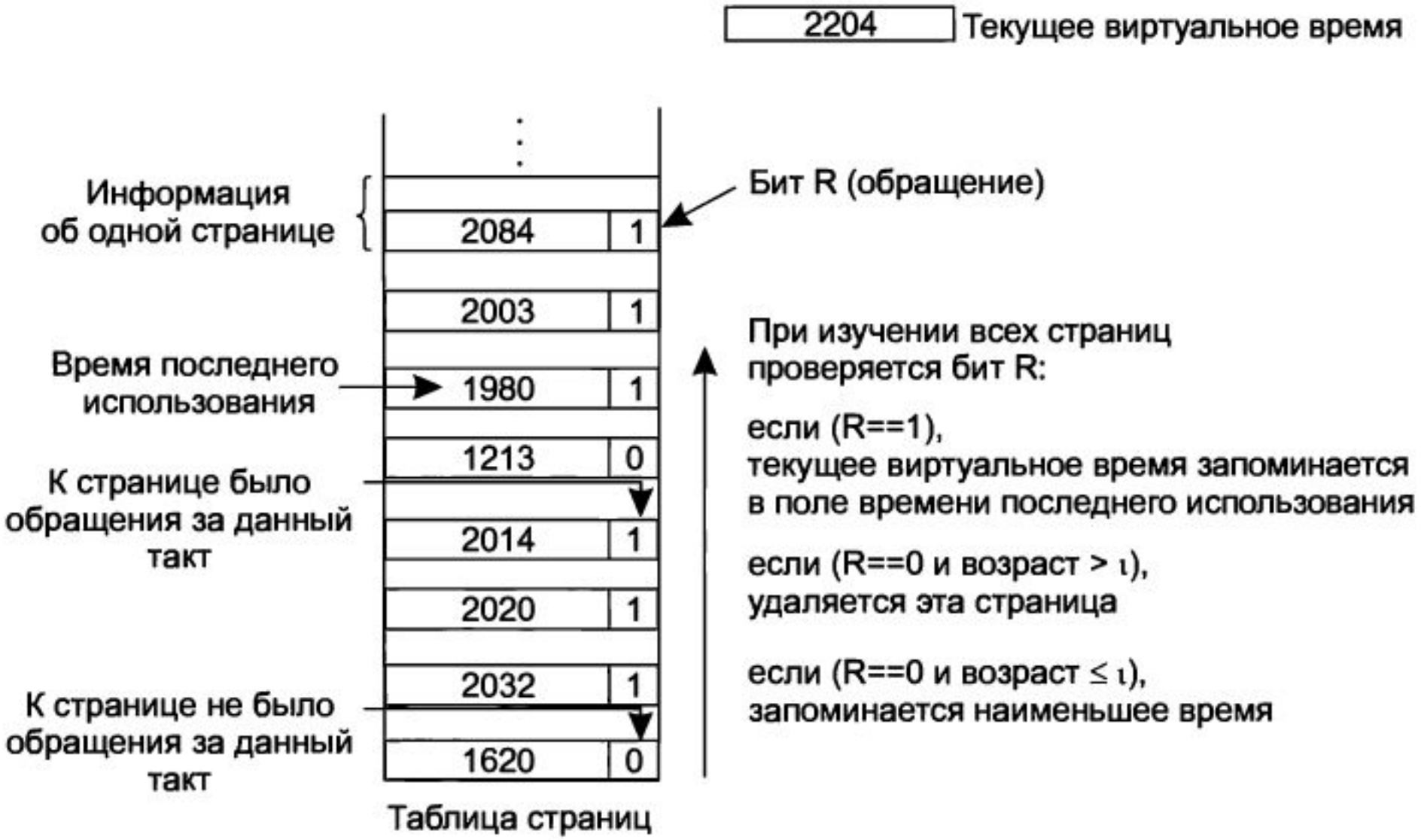
Модификация состоит из двух частей:

1. перед добавлением к счетчикам значения бита R их значение сдвигается на один разряд вправо.
2. значение бита R добавляется к самому левому, а не к самому правому биту.

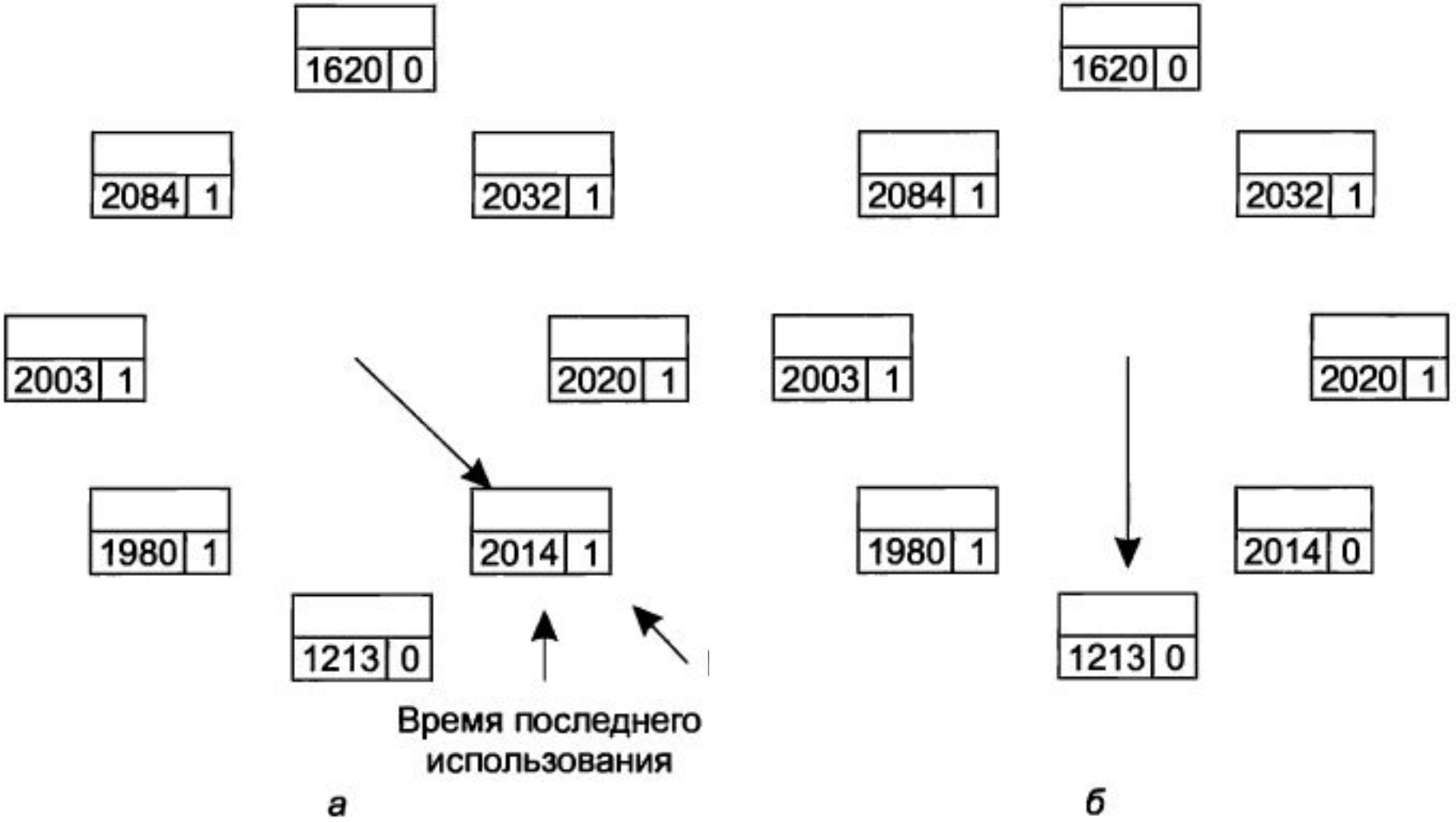

Биты R для страниц 0–5, такт 0
1 0 1 0 1 1
Биты R для страниц 0–5, такт 1
1 1 0 0 1 0
Биты R для страниц 0—5, такт 2
1 1 0 1 0 1
Биты R для страниц 0–5, такт 3
1 0 0 0 1 0
Биты R для страниц 0–5, такт 4
0 1 1 0 0 0
Страница
0 10000000 11000000 11100000 11110000 01111000
1 00000000 10000000 11000000 01100000 10110000
2 10000000 01000000 00100000 00100000 10001000
3 00000000 00000000 10000000 01000000 00100000
4 10000000 11000000 01100000 10110000 01011000
5 10000000 01000000 10100000 01010000 00101000
а
б
в
г
д

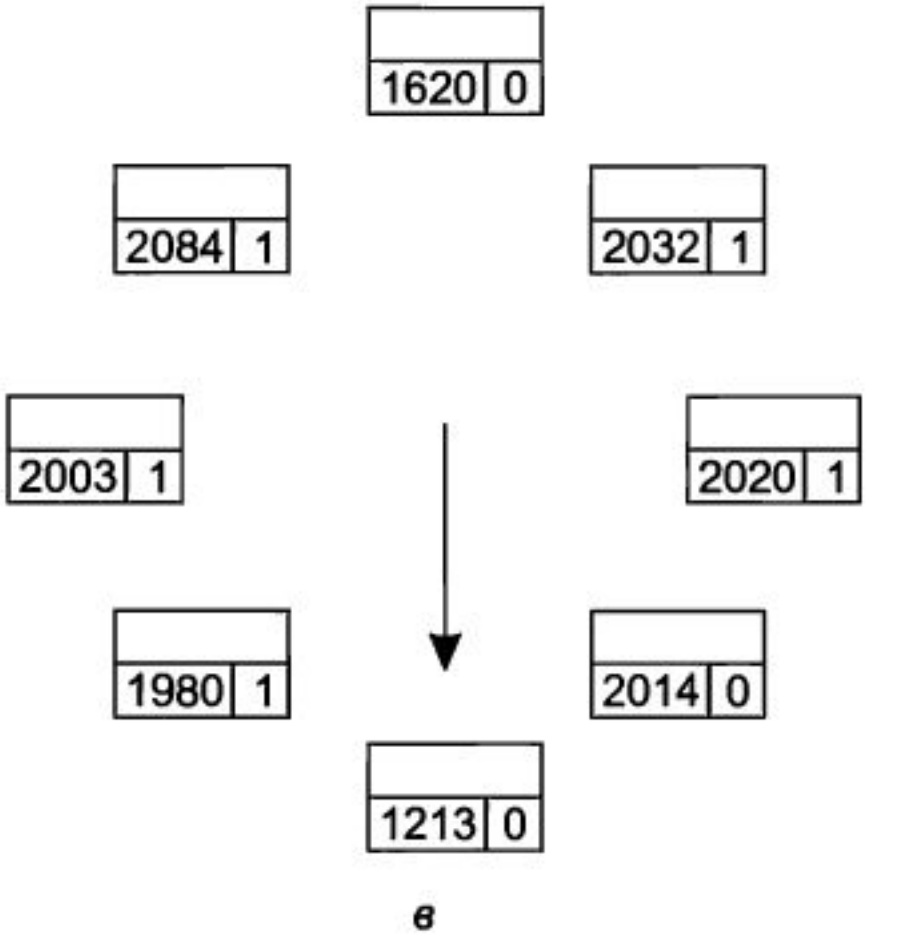
8. Алгоритм «Рабочий набор»

При использовании замещения страниц в простейшей форме процессы начинают свою работу, не имея в памяти вообще никаких страниц. Как только центральный процессор пытается извлечь первую команду, он получает ошибку отсутствия страницы, заставляющую операционную систему ввести в память страницу, содержащую первую команду. Обычно вскоре за этим следуют ошибки отсутствия страниц с глобальными переменными и стеком. Через некоторое время процесс располагает большинством необходимых ему страниц и приступает к работе, сталкиваясь с ошибками отсутствия страниц относительно редко. Эта стратегия называется замещением страниц по требованию (demand paging) поскольку страницы загружаются только по мере надобности, а не заранее.


2204
Текущее виртуальное время
Информация об одной странице
Бит R (обращение)
2084
1
2003
1
Время последнего использования
1980
1
1213
0
К странице было обращения за данный такт
2014
1
2020
1
2032
1
К странице не было обращения за данный такт
1620
0
Таблица страниц
При изучении всех страниц проверяется бит R:
если (R==1), текущее виртуальное время запоминается в поле времени последнего использования
если (R==0 и возраст > ι), удаляется эта страница
если (R==0 и возраст ≤ ι), запоминается наименьшее время

9. Алгоритм WSClock




1620 0
2084 1
2032 1
2003 1
2020 1
1980 1
2014 0
1213 0

в

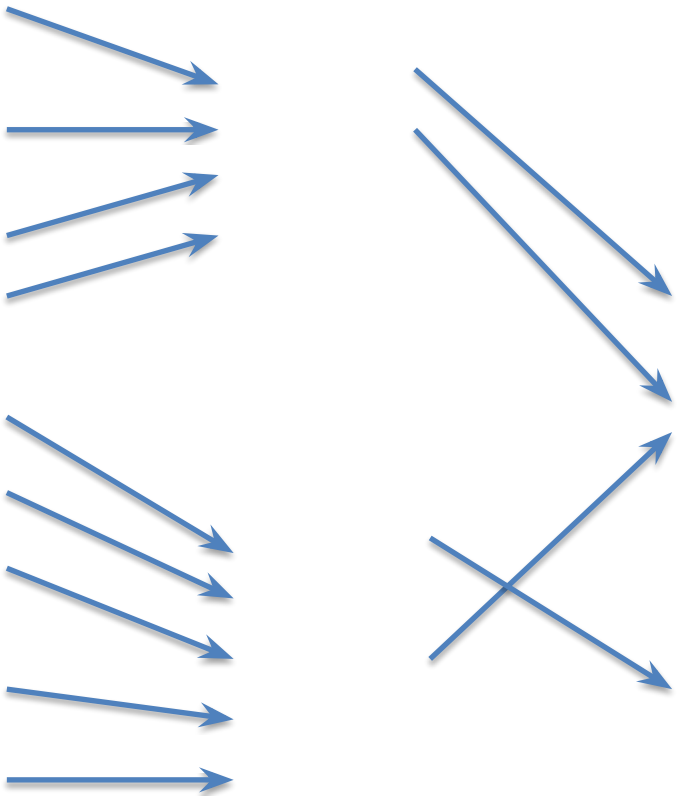

1620 0
2084 1
2032 1
2003 1
2020 1
1980 1
2014 0
2204 1
Новая страница

г

Сравнительная характеристика алгоритмов замещения страниц

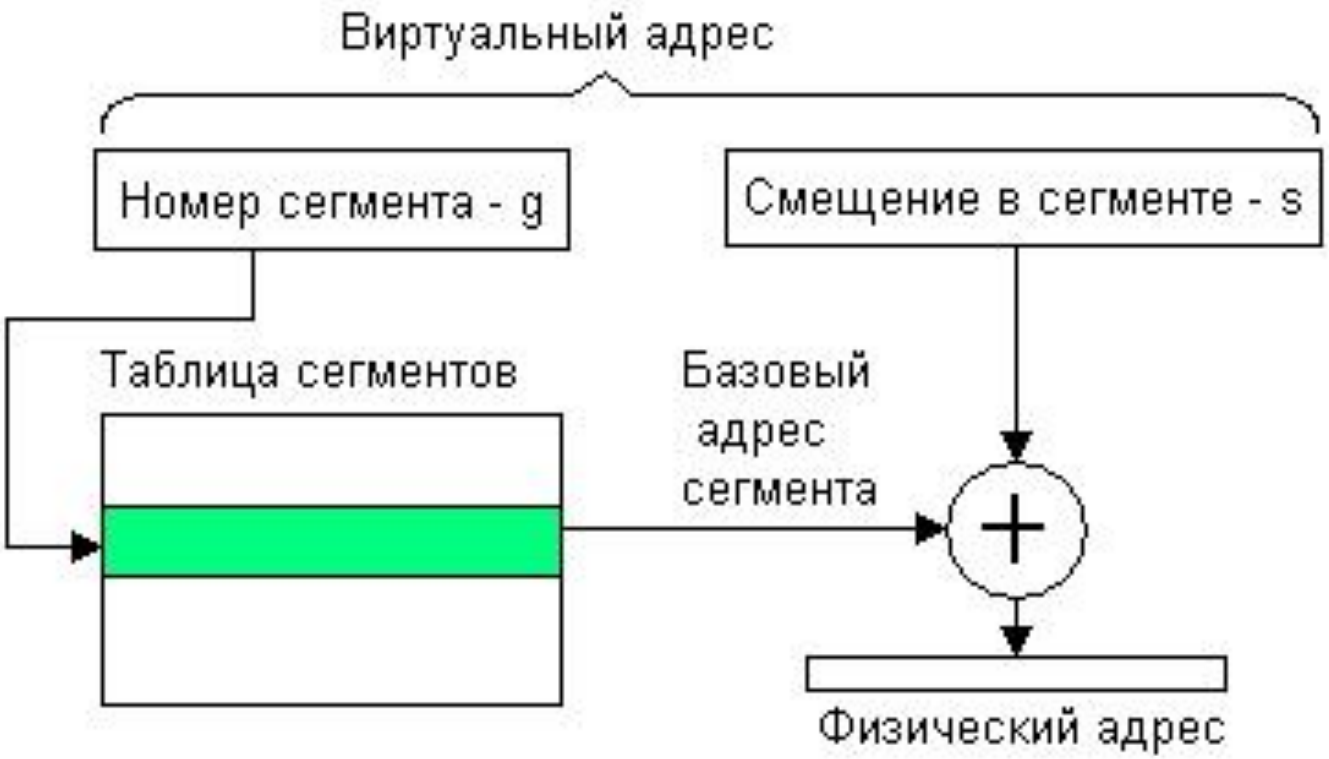
Алгоритм	Особенности
Оптимальный	Не может быть реализован, но полезен в качестве оценочного критерия
NRU (Not Recently Used) — алгоритм исключения недавно использовавшейся страницы	Является довольно грубым приближением к алгоритму LRU
FIFO (First-In, First-Out) — алгоритм «первой пришла, первой и ушла»	Может выгрузить важные страницы
Алгоритм «второй шанс»	Является существенным усовершенствованием алгоритма FIFO
Алгоритм «часы»	Вполне реализуемый алгоритм
LRU (Least Recently Used) — алгоритм замещения наименее востребованной страницы	Очень хороший, но труднореализуемый во всех тонкостях алгоритм
NFU (Not Frequently Used) — алгоритм нечастого востребования	Является довольно грубым приближением к алгоритму LRU
Алгоритм старения	Вполне эффективный алгоритм, являющийся неплохим приближением к алгоритму LRU
Алгоритм рабочего набора	Весьма затратный для реализации алгоритм
WSClock	Вполне эффективный алгоритм

Сегментное распределение памяти



- Формат дескриптора сегмента
  - начальный физический адрес сегмента в оперативной памяти
  - размер сегмента
  - правила доступа
  - признаки модификации, присутствия, обращения к данному сегменту

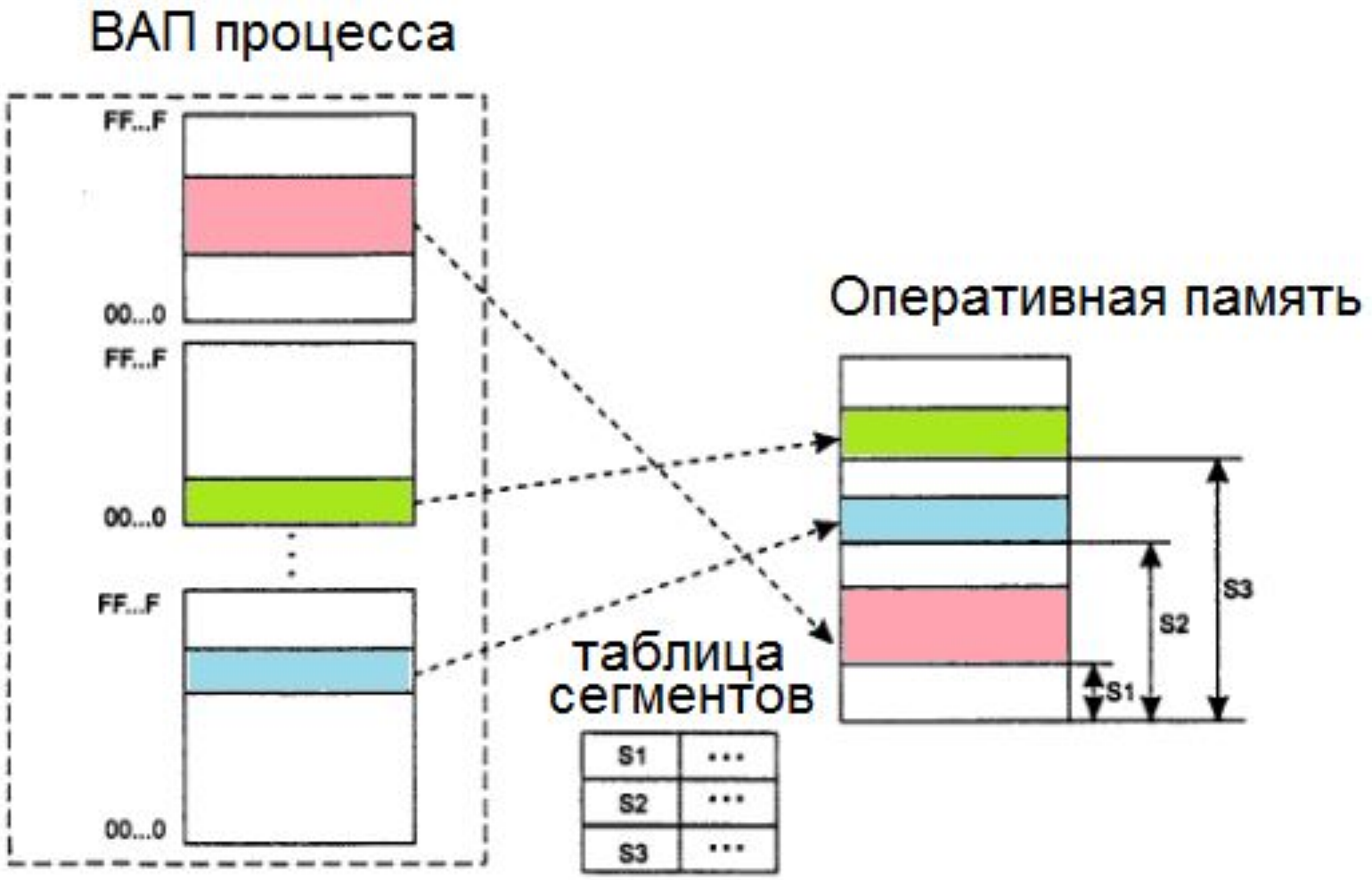
Преобразование виртуального адреса в физический при сегментной организации памяти

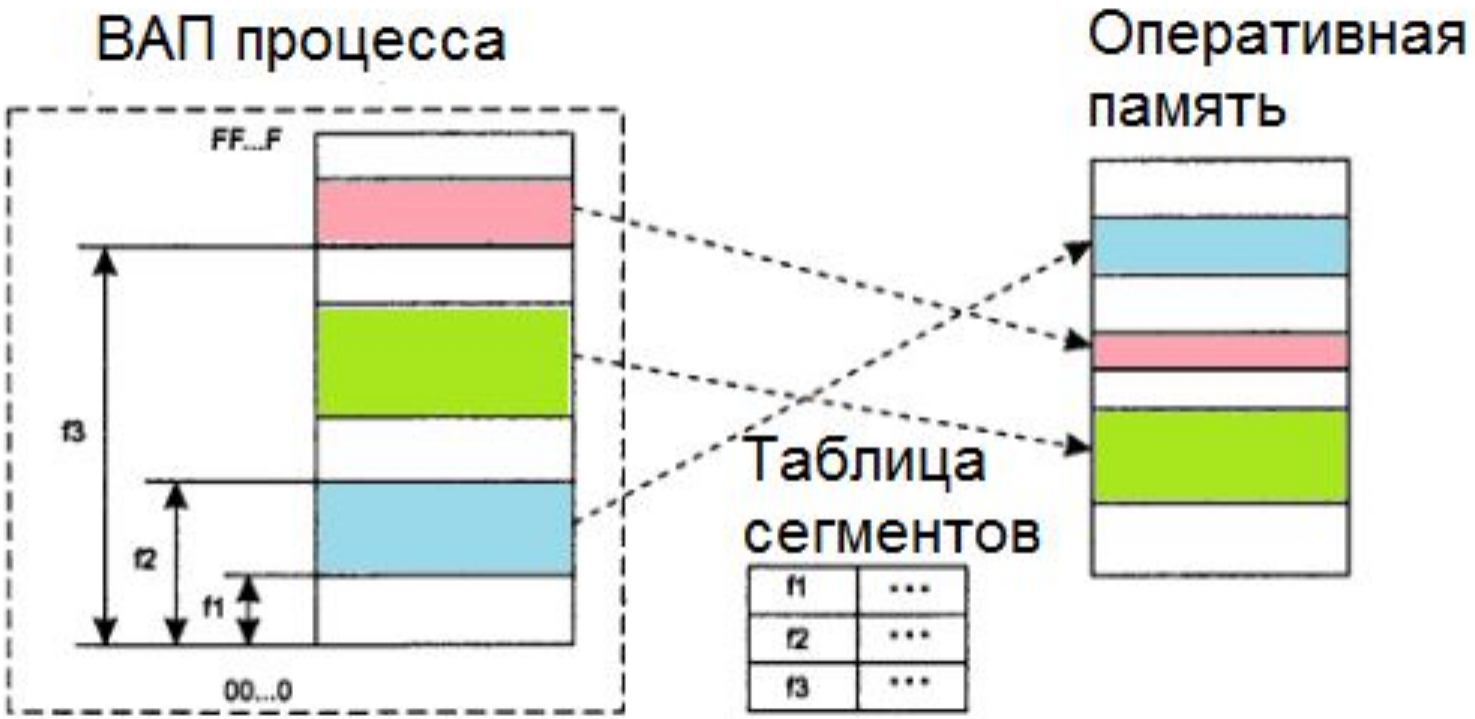


Сегментно-страничное распределение

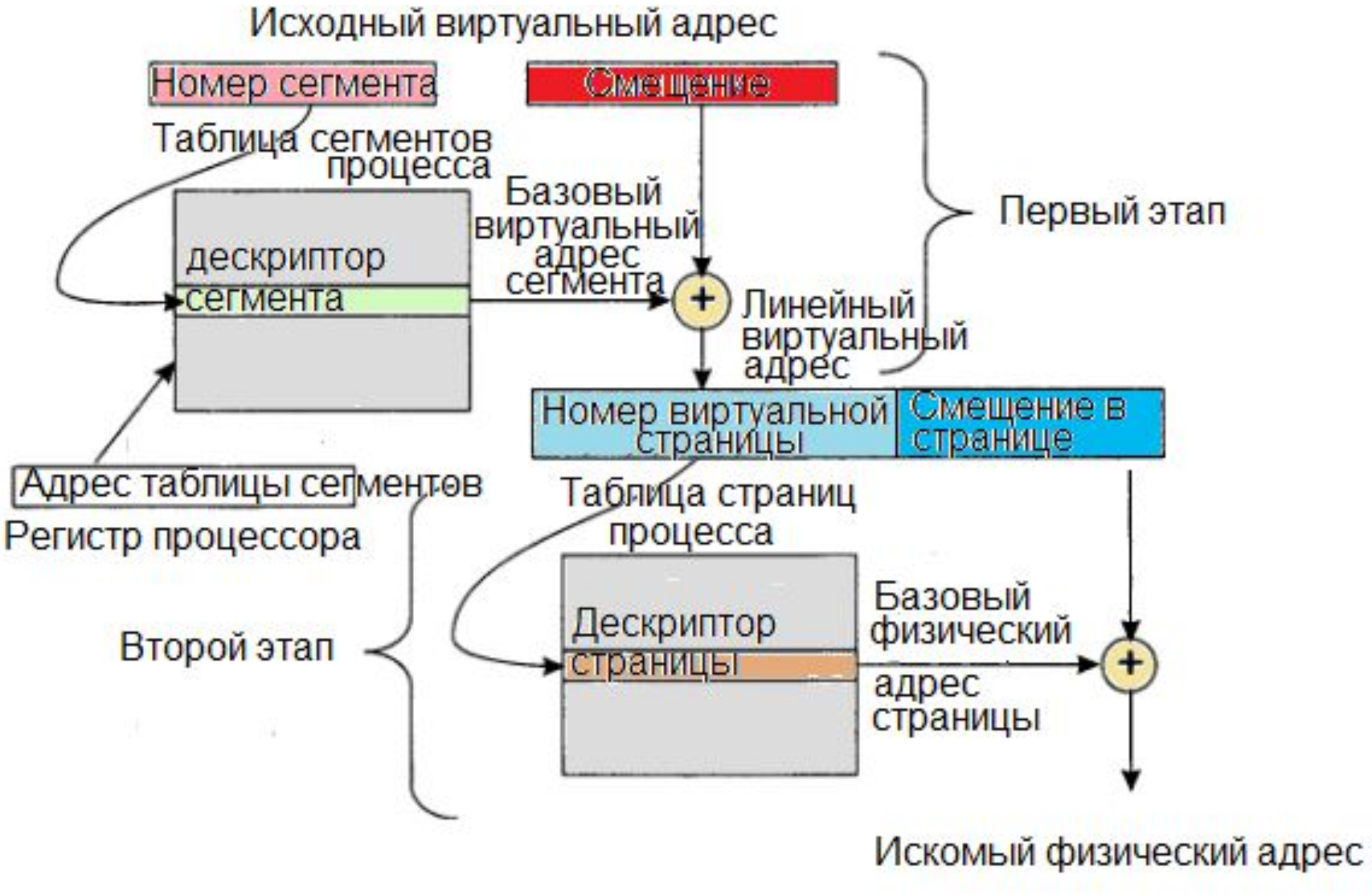
На первом этапе работает механизм сегментации. Исходный виртуальный адрес преобразуется в линейный виртуальный адрес. Для этого на основании базового адреса таблицы сегментов и номера сегмента вычисляется адрес дескриптора сегмента. Проверяются права доступа к сегменту.

страничный механизм. Полученный линейный виртуальный адрес преобразуется в искомый физический адрес. В результате преобразования линейный виртуальный адрес представляется в том виде, в котором он используется при страничной организации памяти (номер страницы, смещение

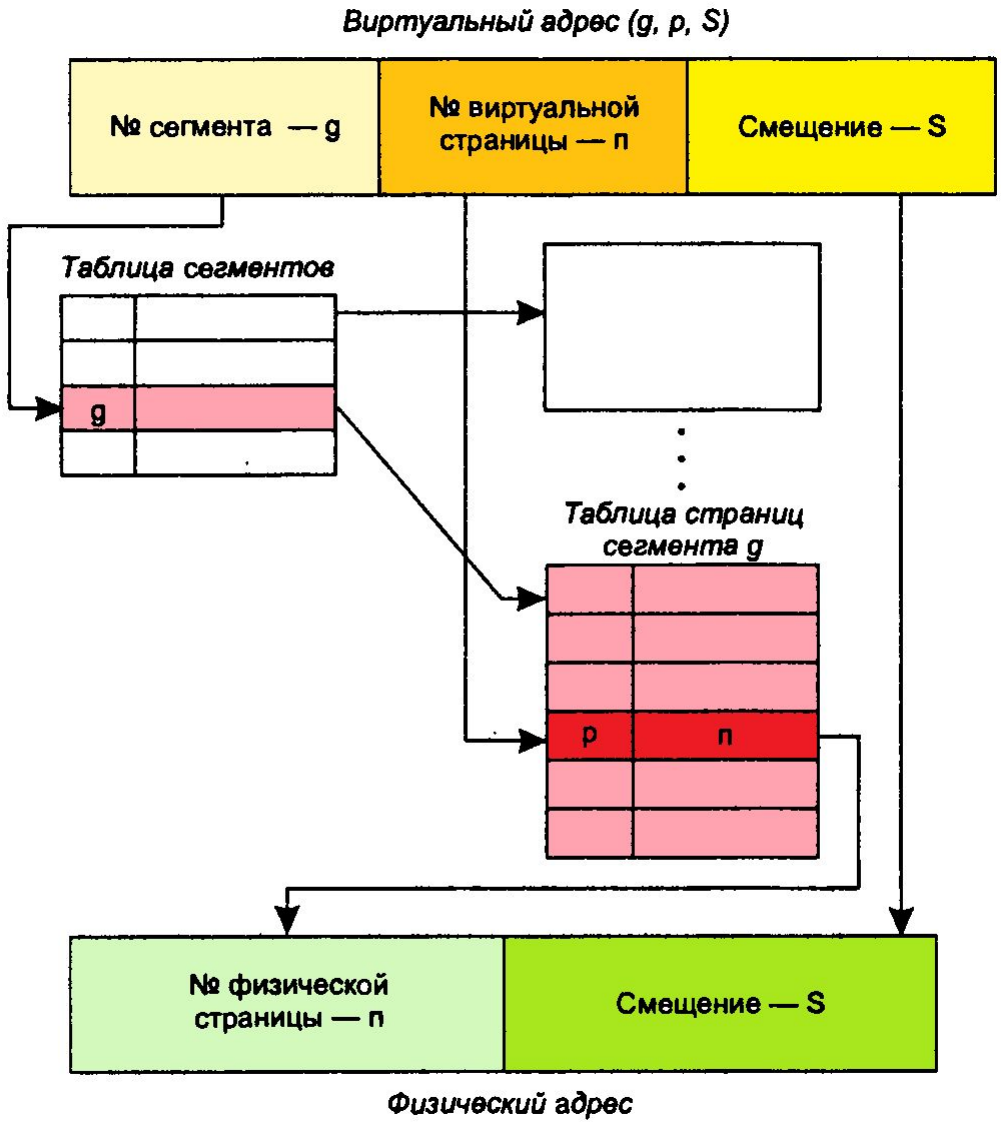

ВАП процесса
FF...F
00...0
FF...F
00...0
FF...F
00...0
Оперативная память
S3
S2
S1
таблица
сегментов
S1 ...
S2 ...
S3 ...


ВАП процесса
FF...F
f3
f2
f1
00...0
Оперативная память
Таблица сегментов
f1 ...
f2 ...
f3 ...

Преобразование виртуального адреса в физический при сегментно-страничной организации памяти

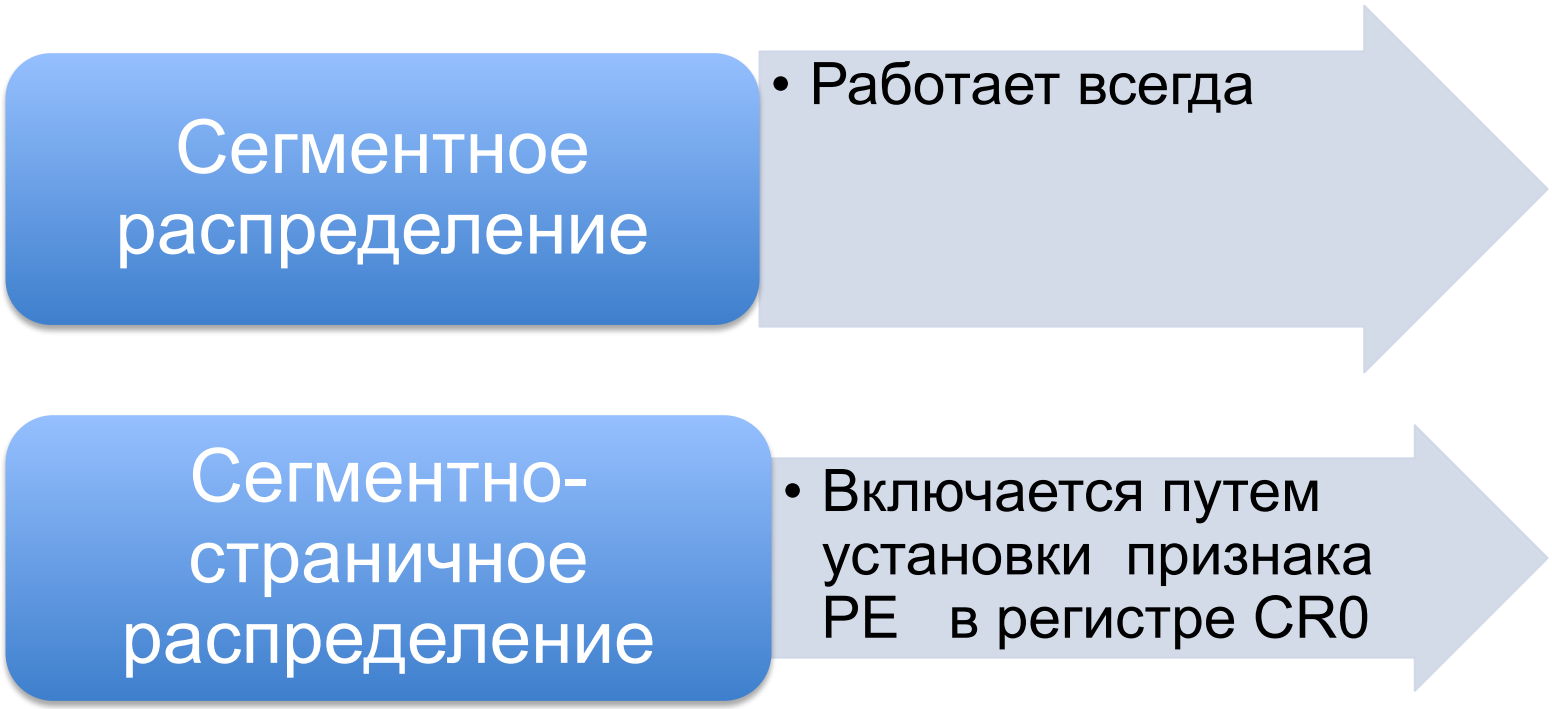


Другая схема сегментно-страничной организации памяти

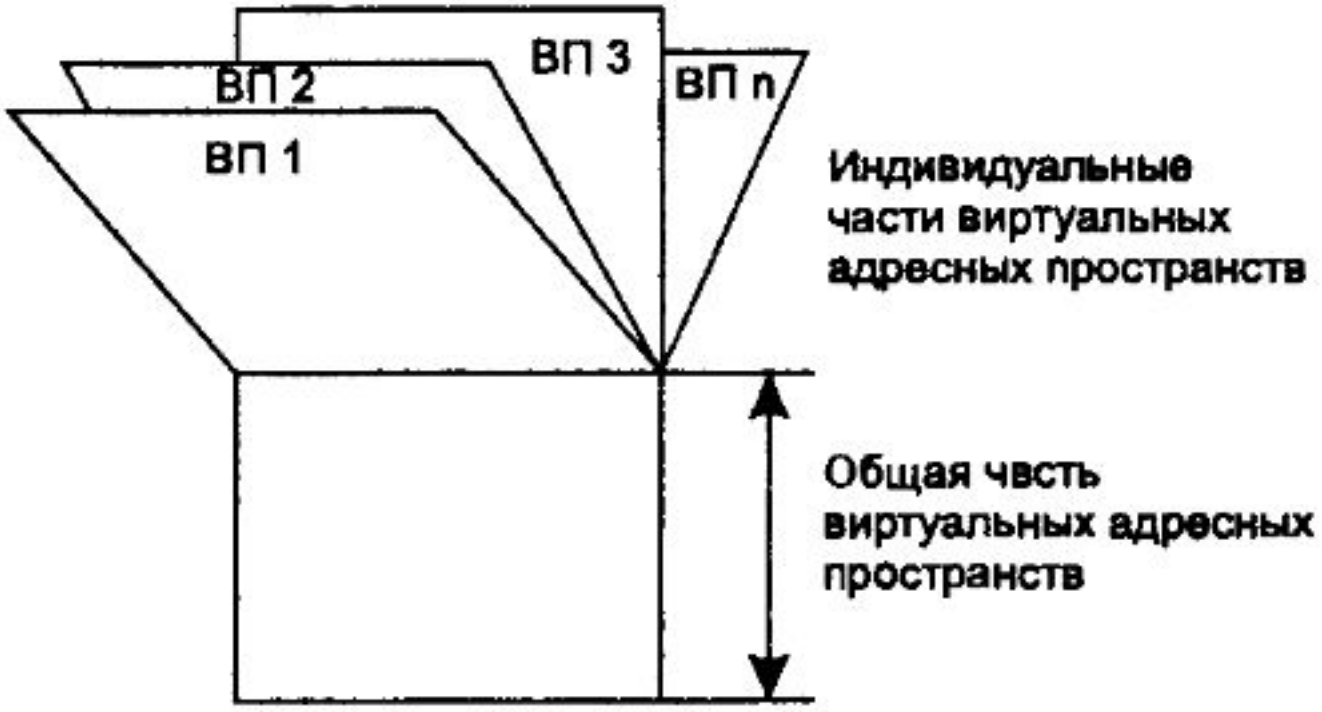


Средства поддержки сегментации памяти в микропроцессоре Intel Pentium

МП поддерживает две модели распределения памяти:



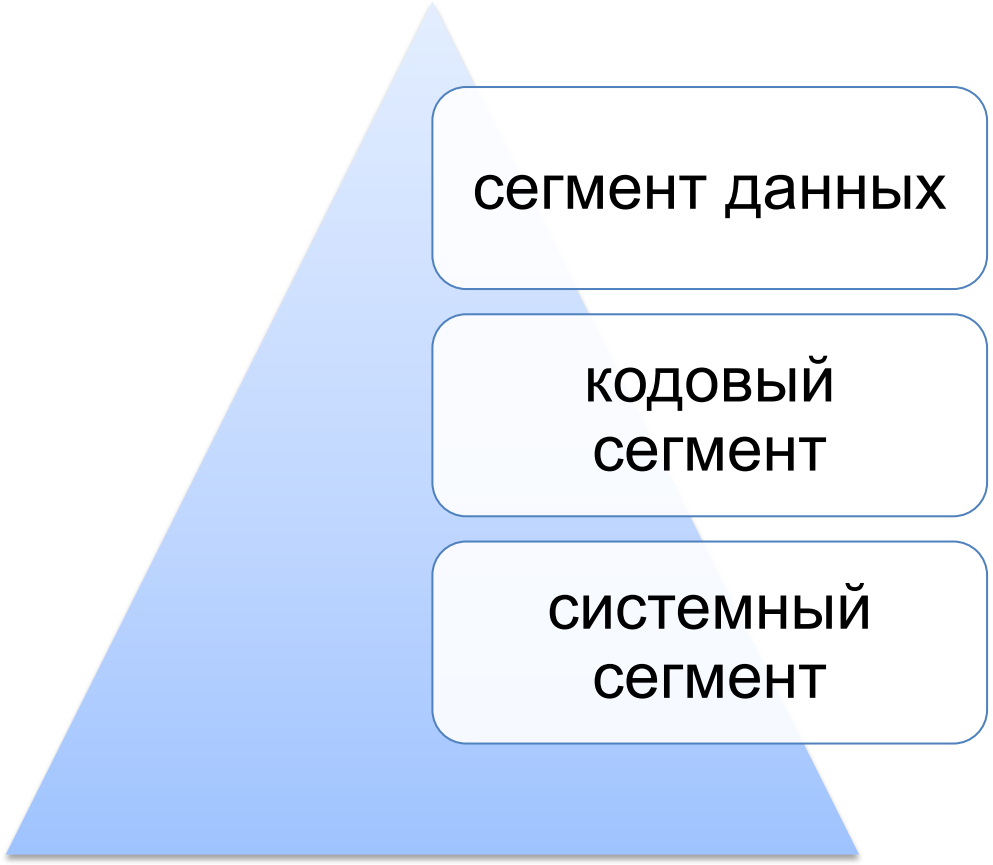
Виртуальное адресное пространство



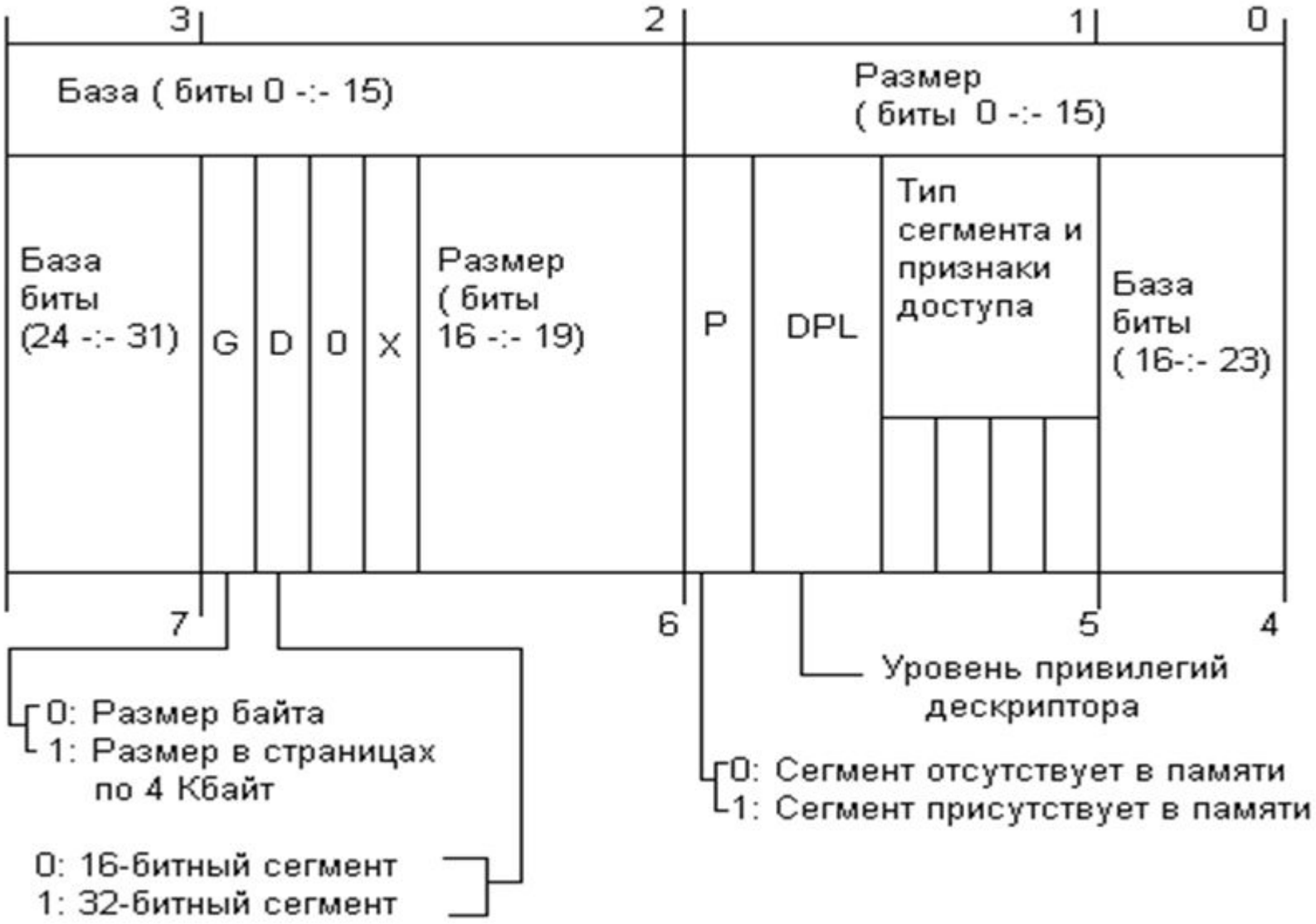
Разрядность поля индекса определяет максимальное число локальных и глобальных дескрипторов процесса - по 8 Кбайт (2^{13}) сегментов каждого типа, всего 16 Кбайт сегментов. Максимальный размер сегмента 4 Гбайт.

Каждый процесс при сегментной организации виртуальной памяти может работать в виртуально адресном пространстве размером 64 Тбайт.

Три основных типа сегментов:



Формат дескриптора сегмента данных или кода

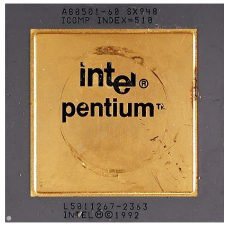


Признаки, задающие тип сегмента и права доступа

МП Intel Pentium поддерживает два типа таблиц дескрипторов сегментов:

1

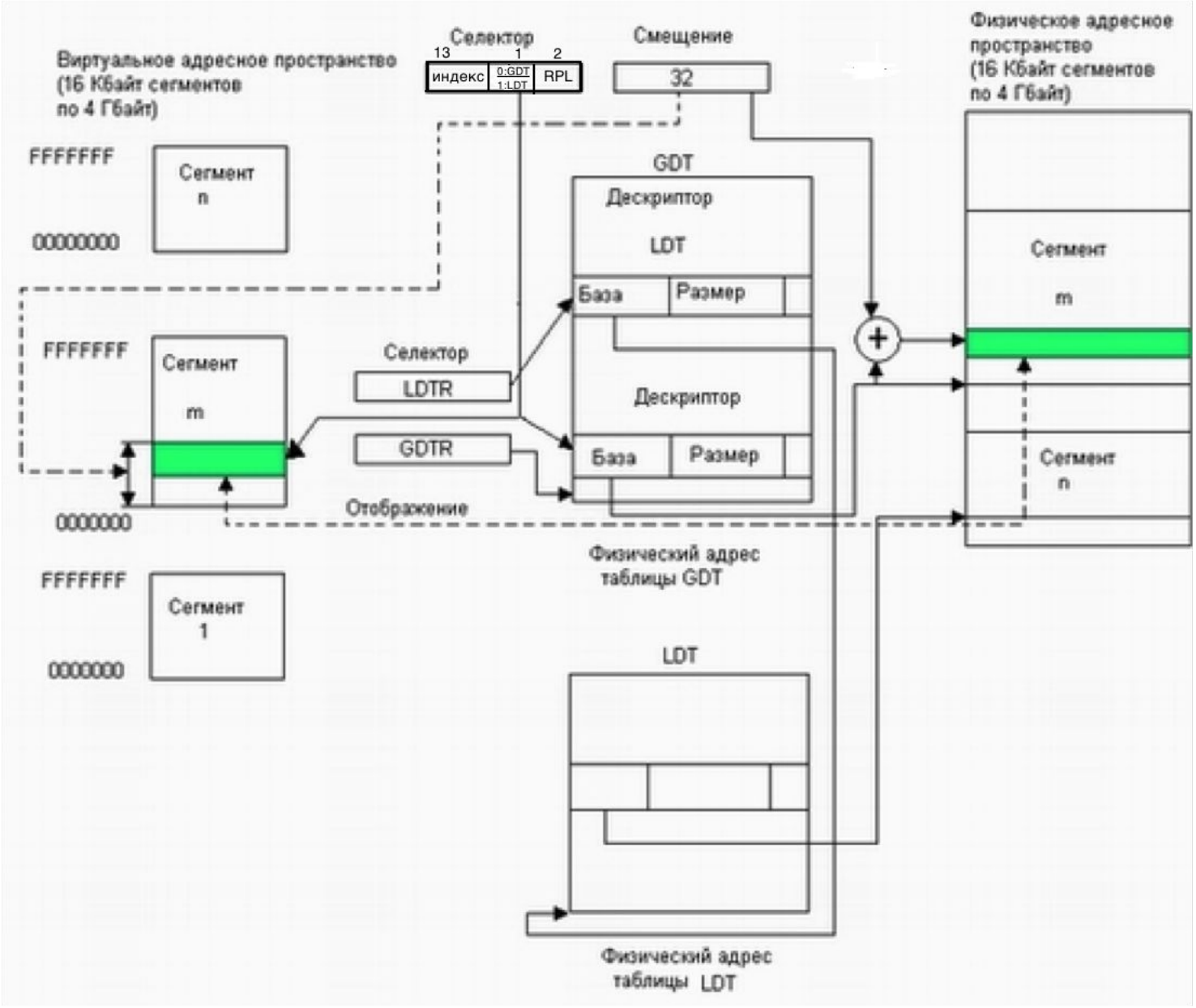
- *глобальная таблица дескрипторов GDT,* предназначена для описания сегментов операционной системы и общих сегментов для всех прикладных процессов, например сегментов межпроцессорного взаимодействия



2

- *локальные таблицы дескрипторов LDT*, которые содержат дескрипторы сегментов отдельных пользовательских процессов

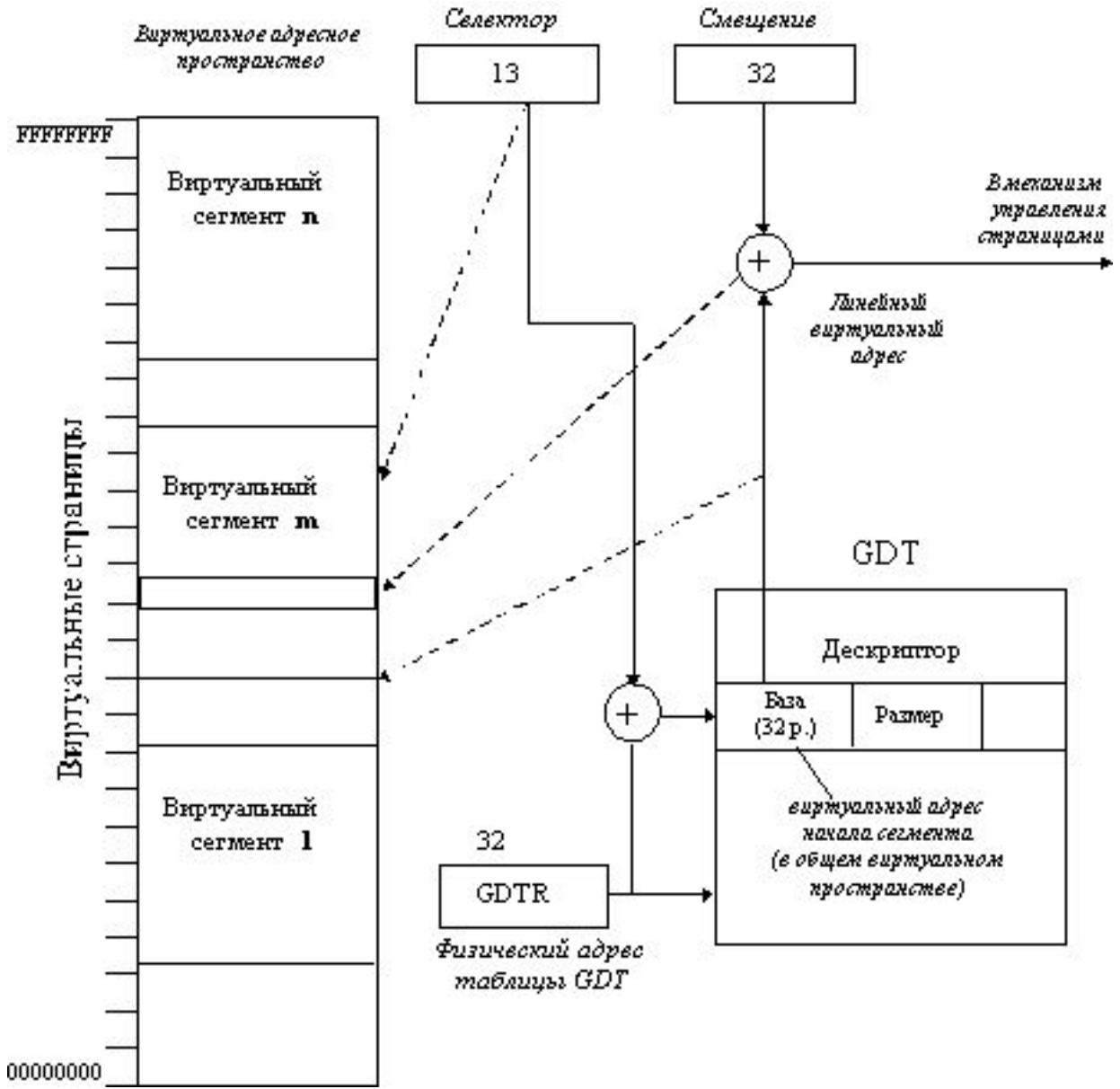
Механизм преобразования виртуального адреса в физический при работе микропроцессора в



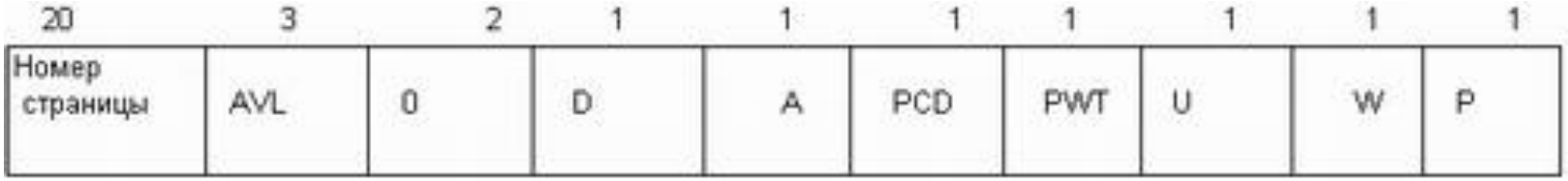
Защита данных при сегментной организации памяти

- ✔ Для каждого процесса поддерживается отдельная таблица дескрипторов LDT
- ✔ Система безопасности на основе привилегий
- ✔ Аппаратные ограничения в наборе инструкций
- ✔ Ограничения на способ использования сегмента

Работа сегментного механизма в сегментно-страничном режиме распределения памяти

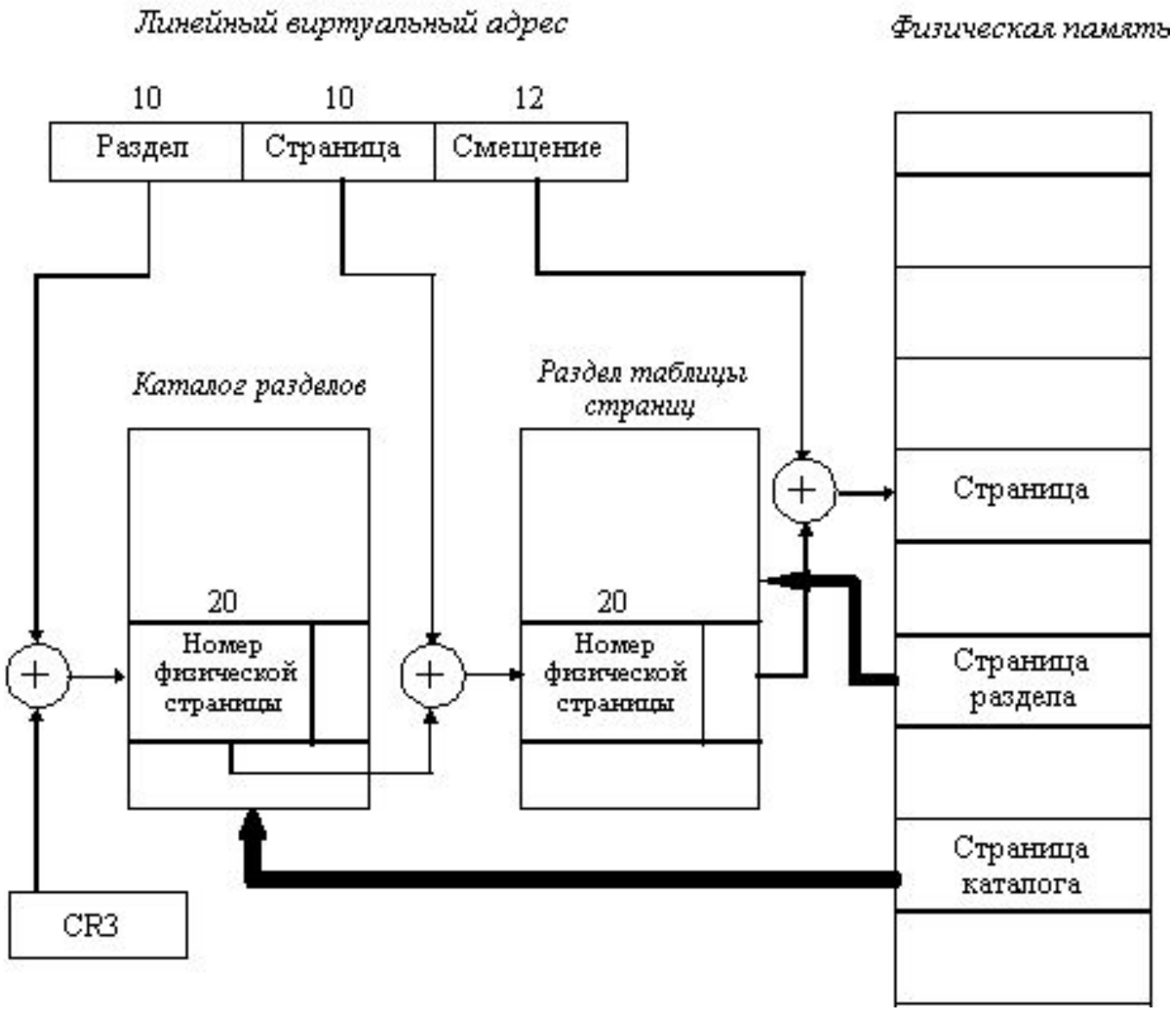


Формат дескриптора страницы



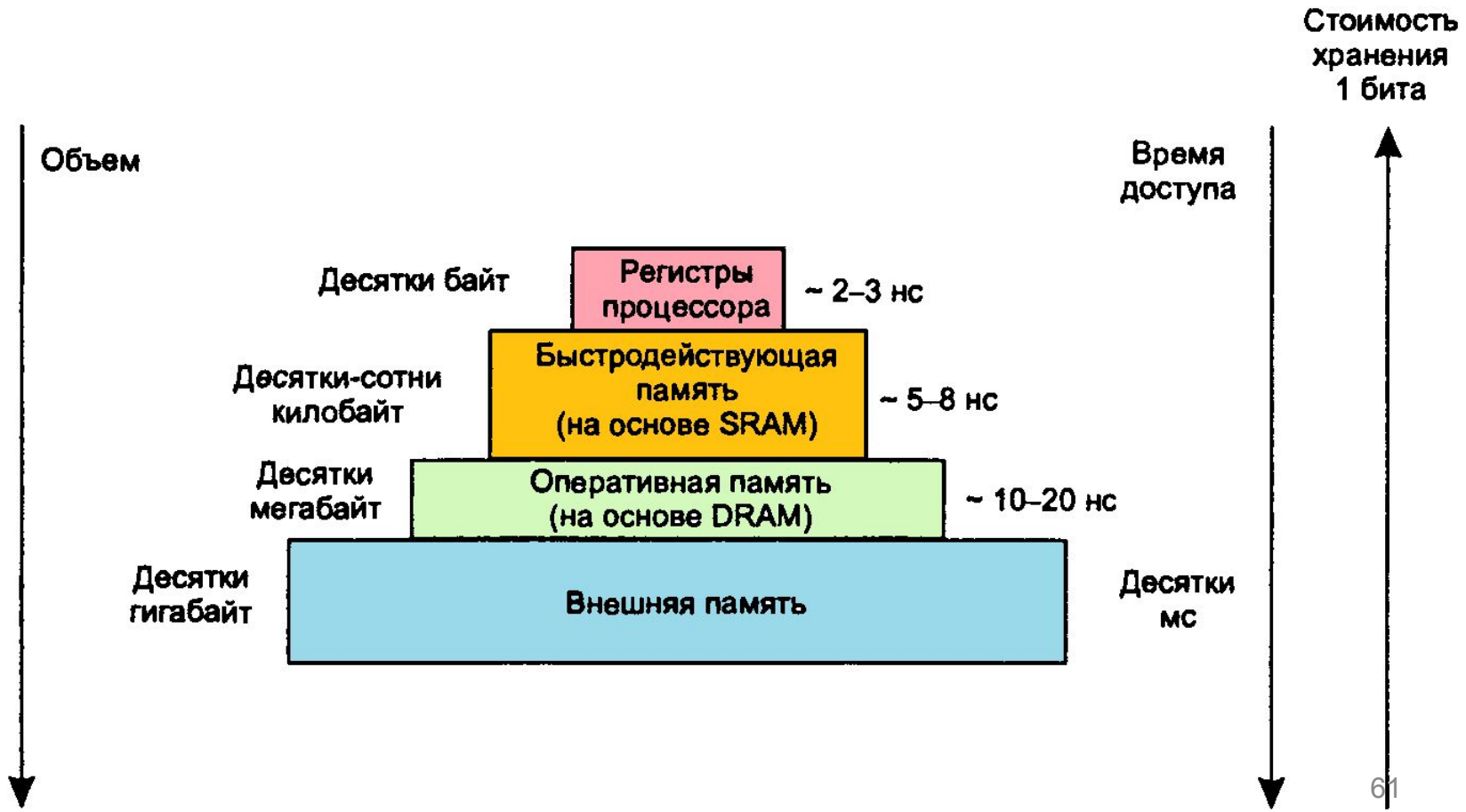
- P — бит присутствия страницы в физической памяти;
- W — бит разрешения записи в страницу;
- U — бит пользователь/супервизор;
- A — признак имевшего место доступа к странице;
- D — признак модификации содержимого страницы;
- PWT и PCD — управляют механизмом кэширования страниц (введены начиная с процессора i486);
- AVL — резерв для нужд операционной системы (AVaiLable for use).

Преобразование линейного виртуального адреса в физический адрес



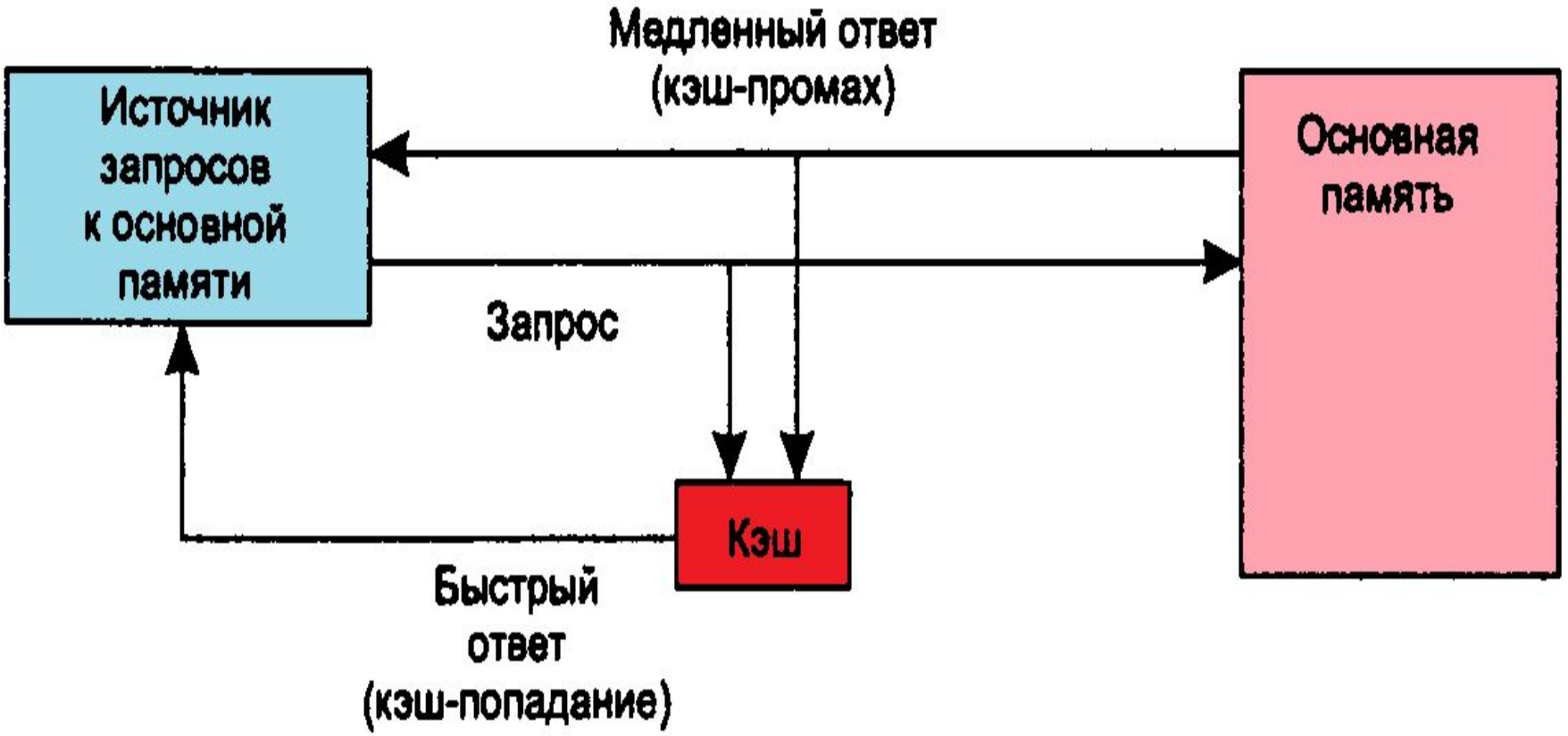
Кэширование данных

Иерархия запоминающих устройств



Кэш – способ совместного функционирования запоминающих устройств, отличающихся временем доступа и стоимостью хранения данных, который за счет динамического копирования в «быстрое» ЗУ наиболее часто используемой информации из «медленного» ЗУ позволяет уменьшить среднее время доступа к данным и экономить более дорогую быстродействующую память.


Адрес данных в основной памяти
Данные
Управляющая информация

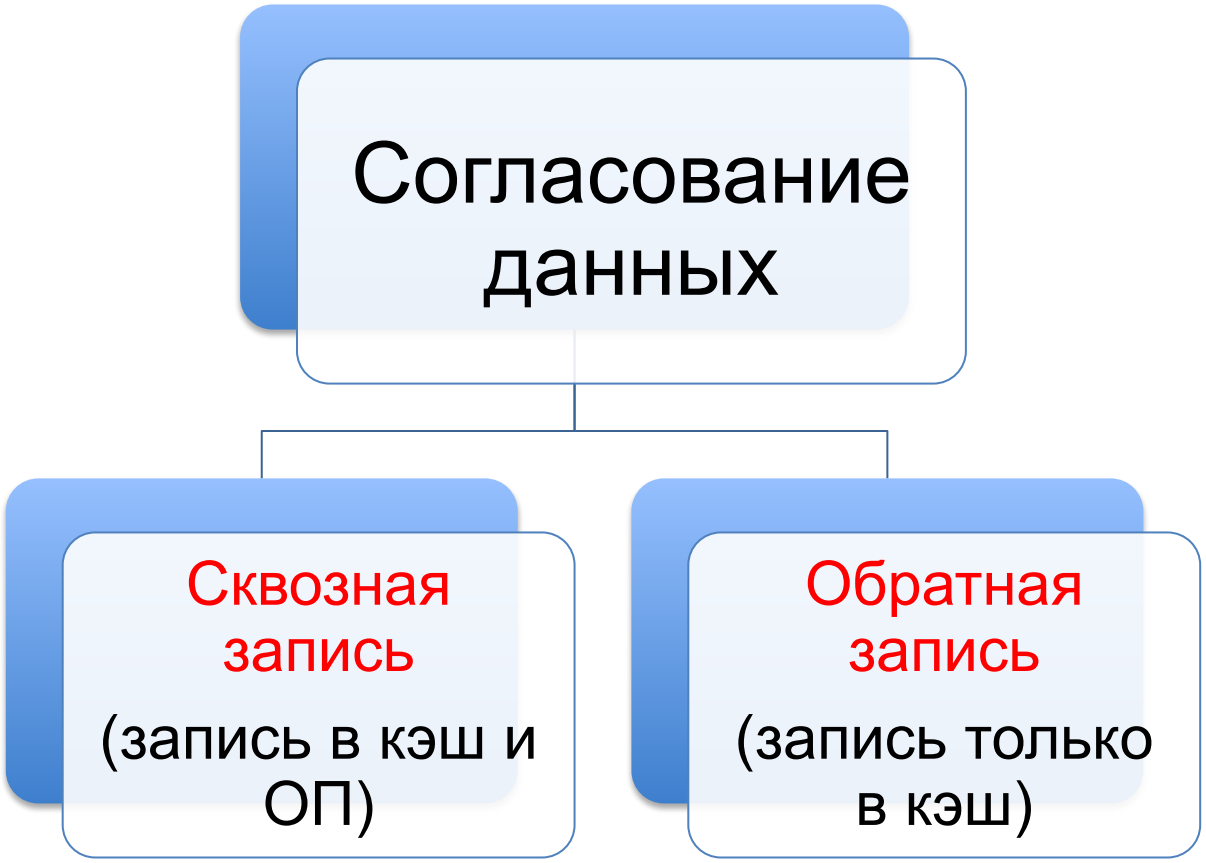

Медленный ответ
(кэш-промах)
Источник запросов к основной памяти
Основная память
Запрос
Кэш
Быстрый ответ
(кэш-попадание)

t_1 - среднее время доступа к основной памяти
t_2 - среднее время доступа к кэш памяти
t - среднее время доступа в системе с кэш-памятью
p –вероятность кэш-попадания

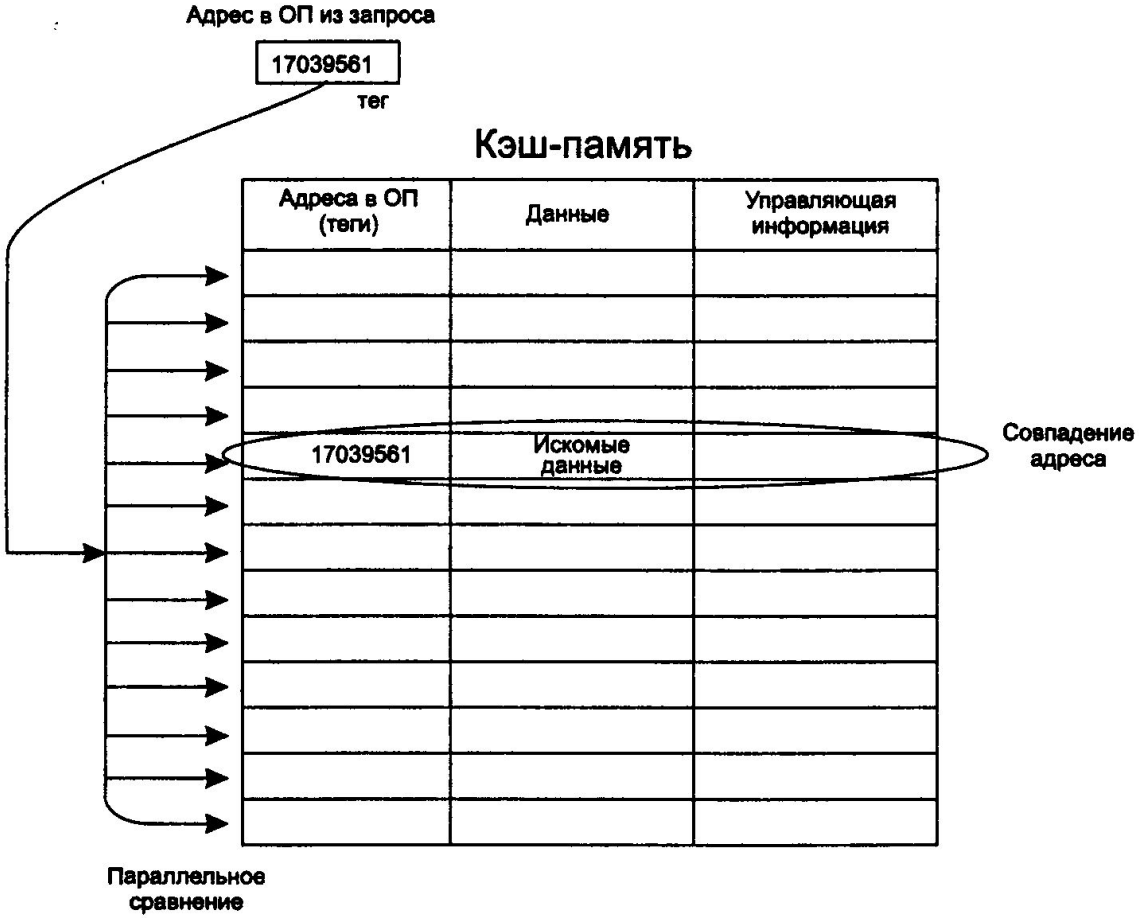
$$t = t_1 (1-p) + t_2 p = (t_2 - t_1)p + t_1$$

Временная локальность. Если произошло обращение по некоторому адресу, то следующее обращение по тому же адресу с большой вероятностью произойдет в ближайшее время.

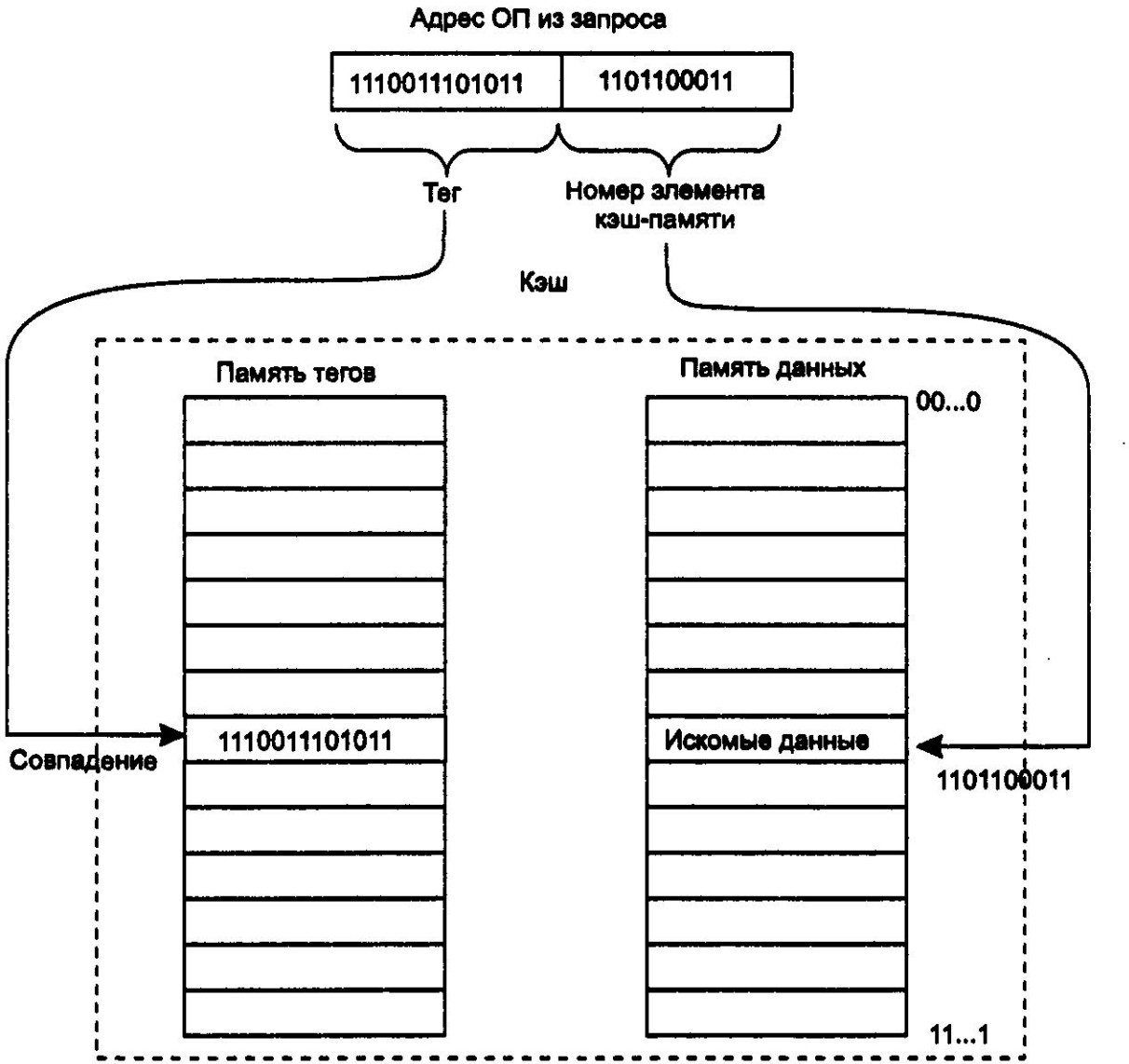
Пространственная локальность. Если произошло обращение по некоторому адресу, то с высокой степенью вероятности в ближайшее время произойдет обращение к соседним адресам.


Согласование данных
Сквозная запись
(запись в кэш и ОП)
Обратная запись
(запись только в кэш)

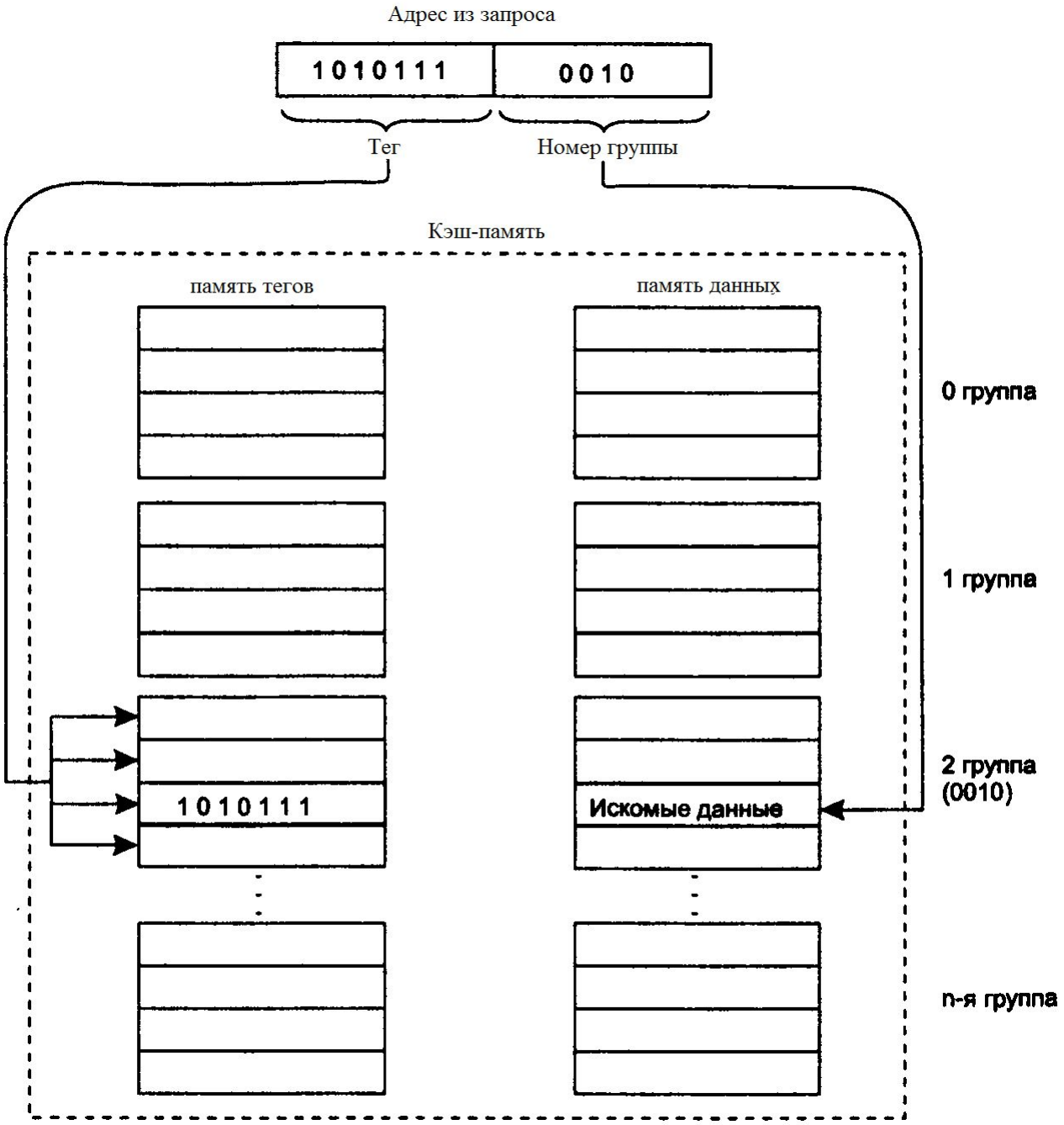
Случайное отображение основной памяти на кэш

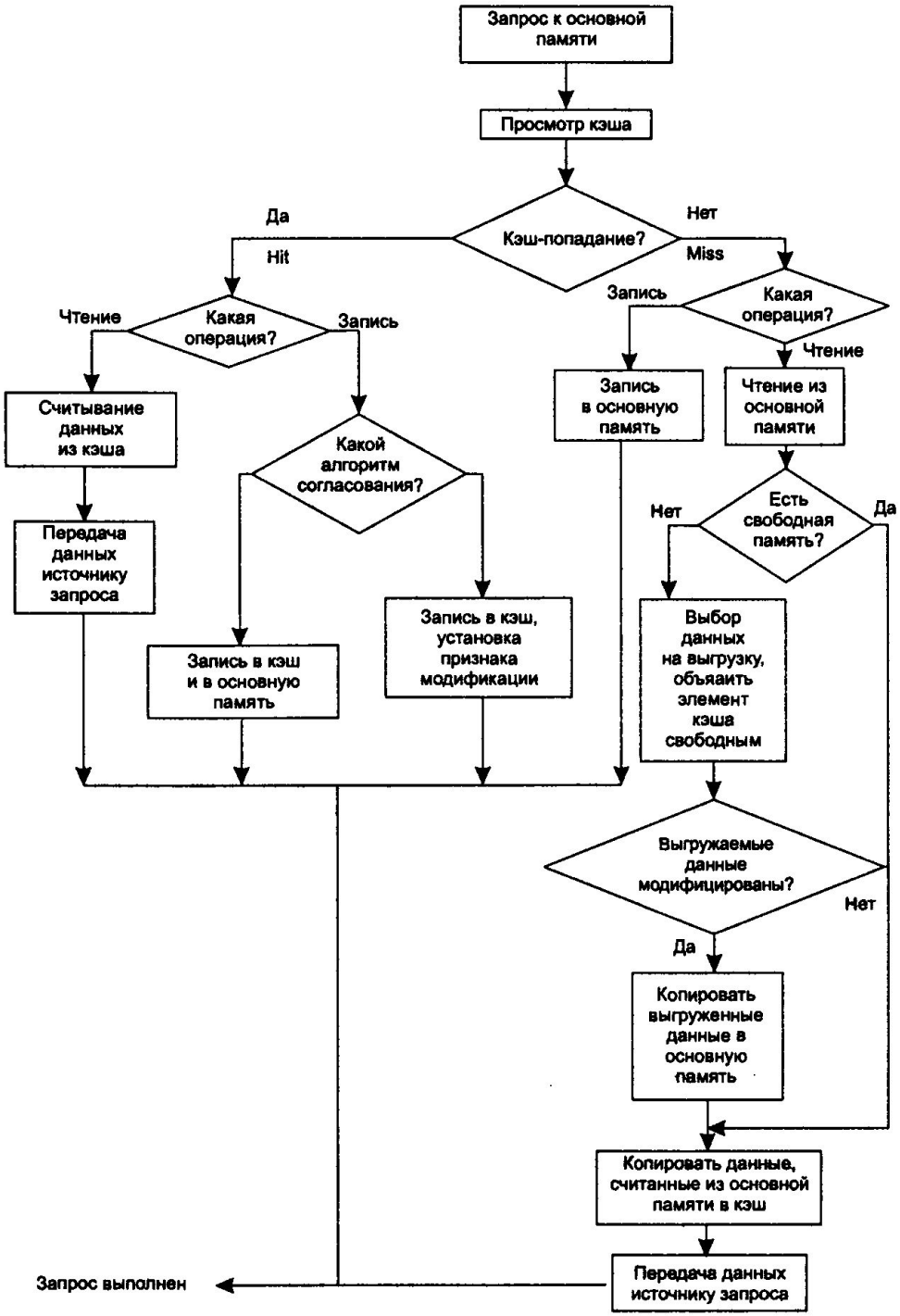


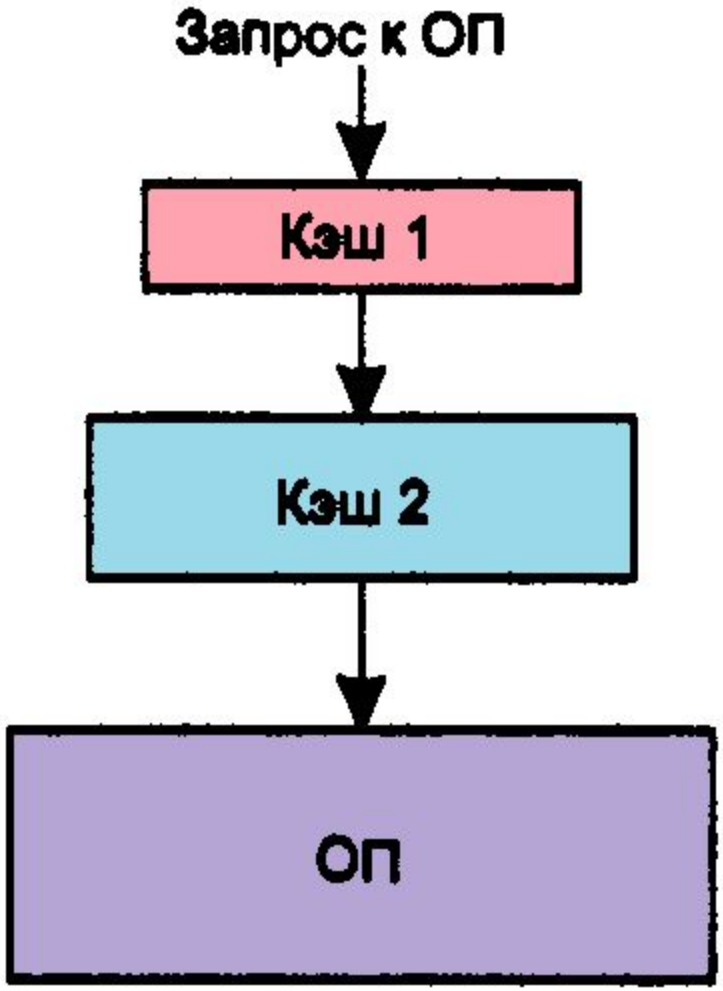
Детерминированное отображение основной памяти на КЭШ

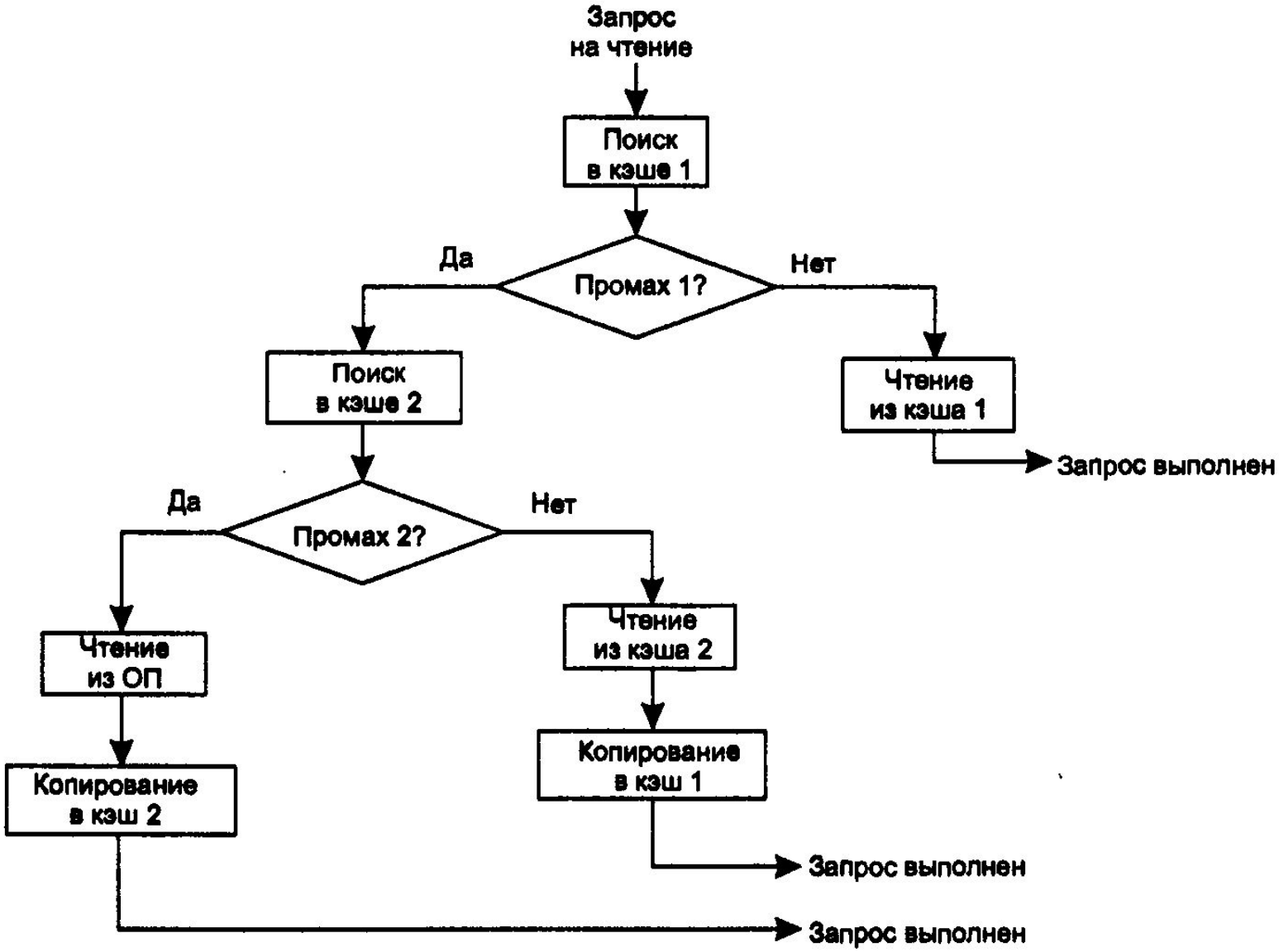


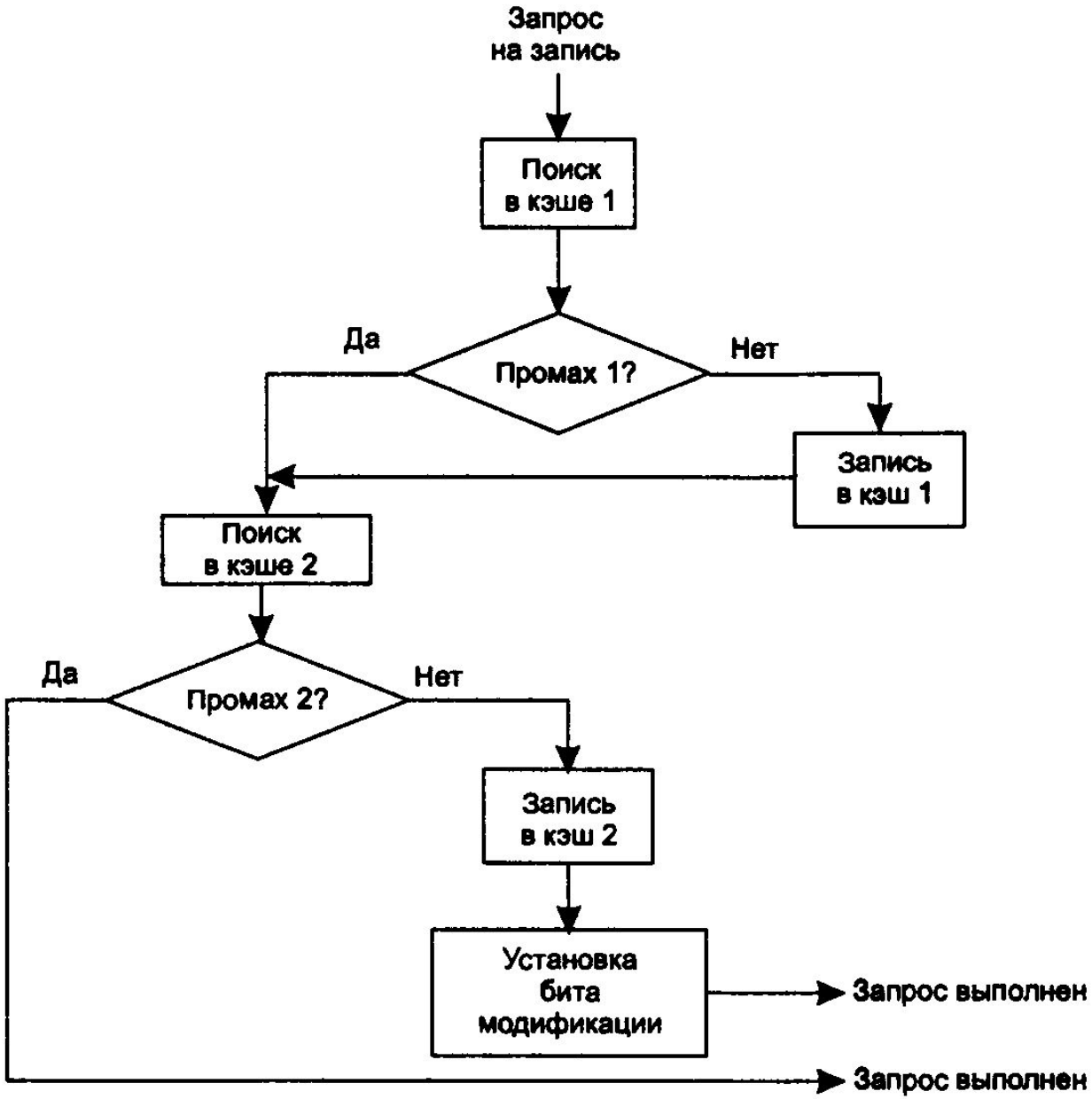
Комбинированный способ




Запрос к основной памяти
Просмотр кэша
Кэш-попадание?
Да
Hit
Нет
Miss
Какая операция?
Чтение
Запись
Какая операция?
Чтение
Запись
Считывание данных из кэша
Какой алгоритм согласования?
Запись в основную память
Чтение из основной памяти
Передача данных источнику запроса
Запись в кэш и в основную память
Запись в кэш, установка признака модификации
Есть свободная память?
Нет
Да
Выбор данных на выгрузку, объяаить элемент кэша свободным
Выгружаемые данные модифицированы?
Нет
Да
Копировать выгруженные данные в основную память
Копировать данные, считанные из основной памяти в кэш
Передача данных источнику запроса
Запрос выполнен


Запрос к ОП
Кэш 1
Кэш 2
ОП


Запрос
на чтение
Поиск
в кэше 1
Промах 1?
Да
Нет
Поиск
в кэше 2
Чтение
из кэша 1
Запрос выполнен
Промах 2?
Да
Нет
Чтение
из ОП
Чтение
из кэша 2
Копирование
в кэш 2
Копирование
в кэш 1
Запрос выполнен
Запрос выполнен


Запрос
на запись
Поиск
в кэше 1
Да
Промах 1?
Нет
Запись
в кэш 1
Поиск
в кэше 2
Да
Промах 2?
Нет
Запись
в кэш 2
Установка
бита
модификации
Запрос выполнен
Запрос выполнен

Уровни кэша

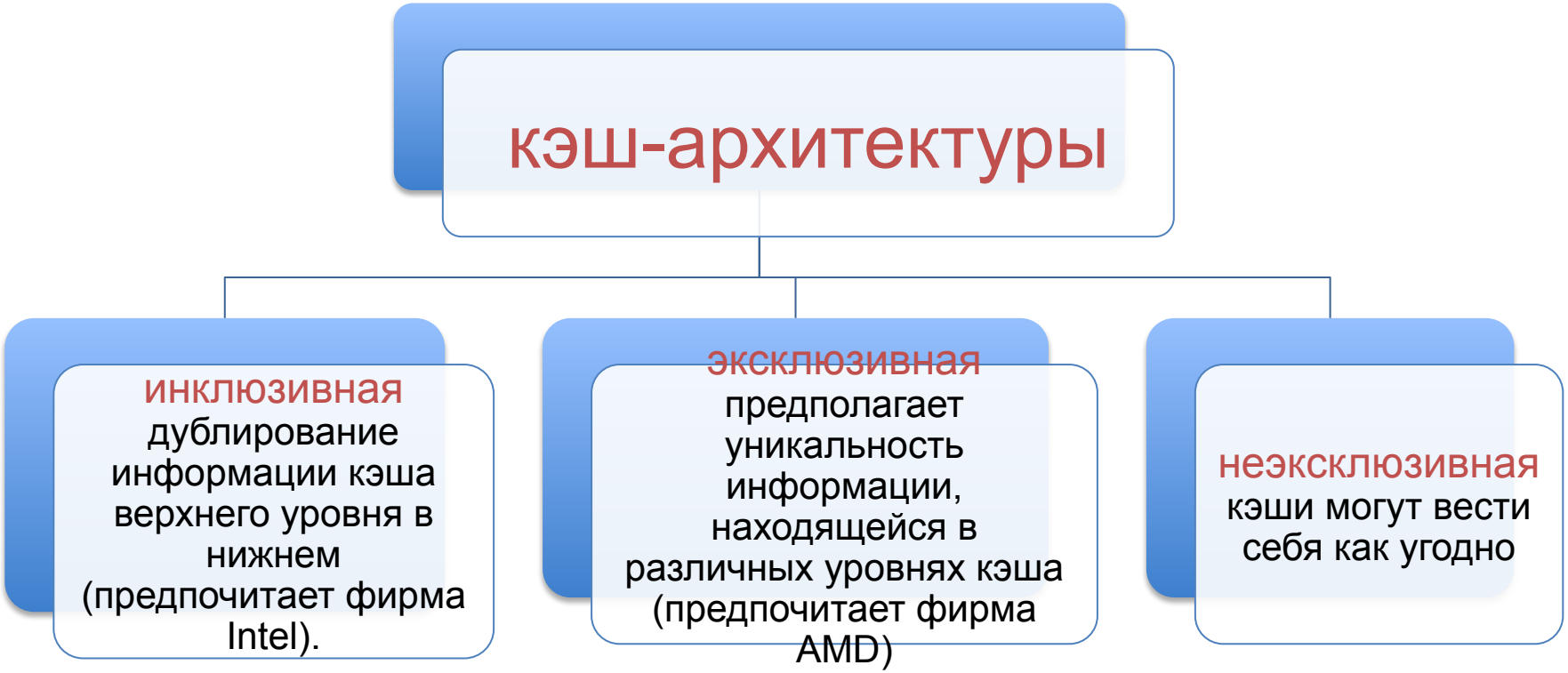
L1-cache.Самая быстрая память. Является неотъемлемой частью процессора, поскольку расположена на одном с ним кристалле и входит в состав функциональных блоков. В современных процессорах обычно кэш L1 разделен на два кэша, кэш команд (инструкций) и кэш данных (Гарвардская архитектура). Большинство процессоров без L1 кэша не могут функционировать. L1 кэш работает на частоте процессора, и, в общем случае, обращение к нему может производиться каждый такт. Зачастую является возможным выполнять несколько операций чтения/записи одновременно. Латентность доступа обычно равна 2−4 тактам ядра. Объём обычно невелик — не более 128 Кбайт.

L2-cache — кэш второго уровня, Второй по быстродействию. Обычно он расположен на кристалле, как и L1. В старых процессорах — набор микросхем на системной плате. Объём L2 кэша от 128 Кбайт до 1−12 Мбайт. В современных многоядерных процессорах кэш второго уровня, находясь на том же кристалле, является памятью раздельного пользования — при общем объёме кэша в nM Мбайт на каждое ядро приходится по nM/nC Мбайта, где nC количество ядер процессора. Обычно латентность L2 кэша, расположенного на кристалле ядра, составляет от 8 до 20 тактов ядра.

L3-cache Кэш третьего уровня наименее быстродействующий, но он может быть очень внушительного размера — более 24 Мбайт. L3 кэш медленнее предыдущих кэшей, но всё равно значительно быстрее, чем оперативная память. В многопроцессорных системах находится в общем пользовании и предназначен для синхронизации данных различных L2.

Иногда существует и 4 уровень кэша, обыкновенно он расположен в отдельной микросхеме. Применение кэша 4 уровня оправдано только для высоко производительных серверов и мейнфреймов.

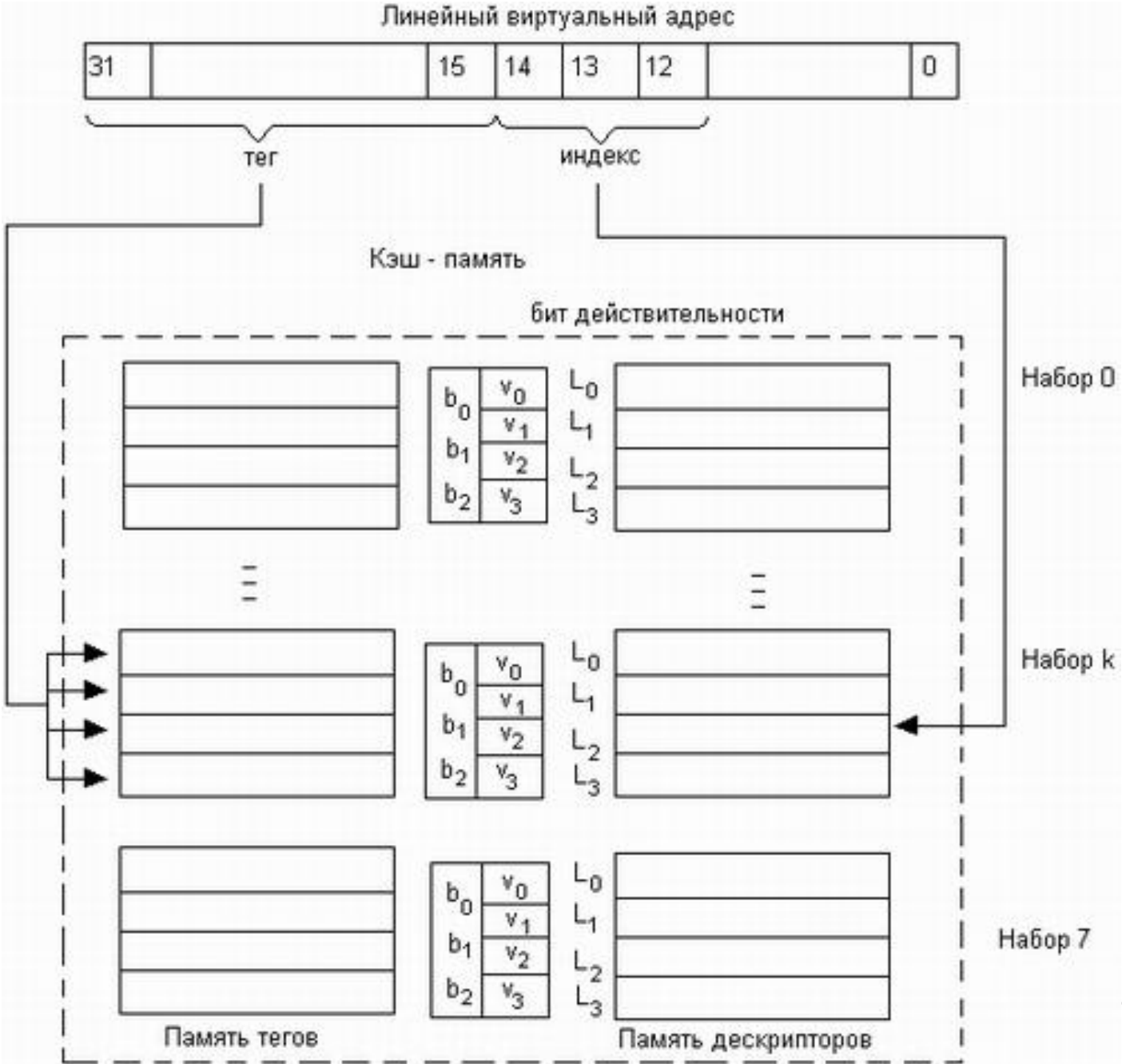
Проблема синхронизации между различными кэшами



Кэширование в МП Intel Pentium

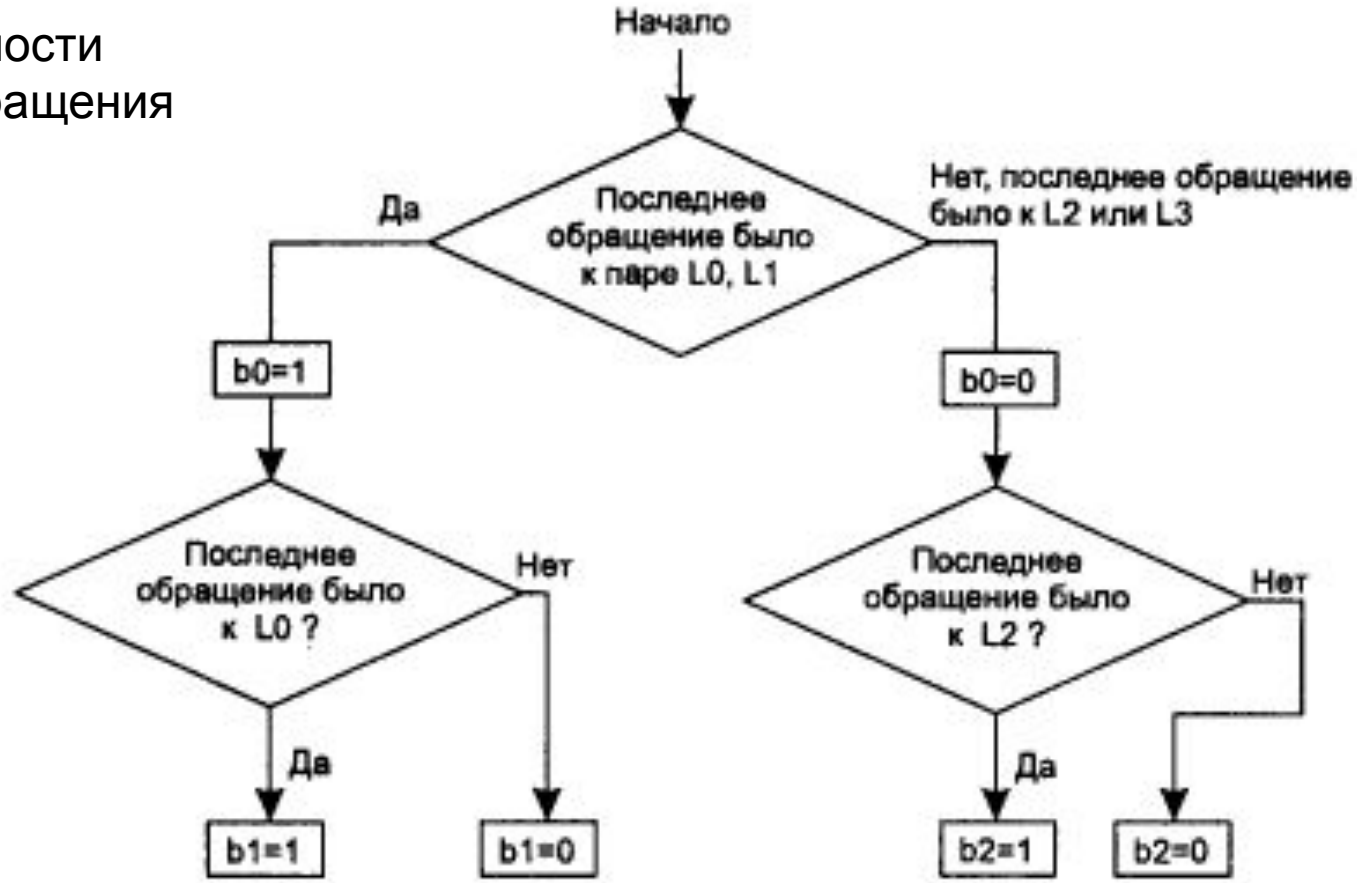
- Кэширование дескрипторов сегментов в скрытых регистрах
- Кэширование пар номеров виртуальных и физических страниц в буфере ассоциативной трансляции TLB
- Кэширование данных и инструкций в кэш-памяти первого уровня
- Кэширование данных и инструкций в кэш-памяти второго уровня

Буфер ассоциативной трансляции



Алгоритм Pseudo LRU (Pseudo Least Recently Used)

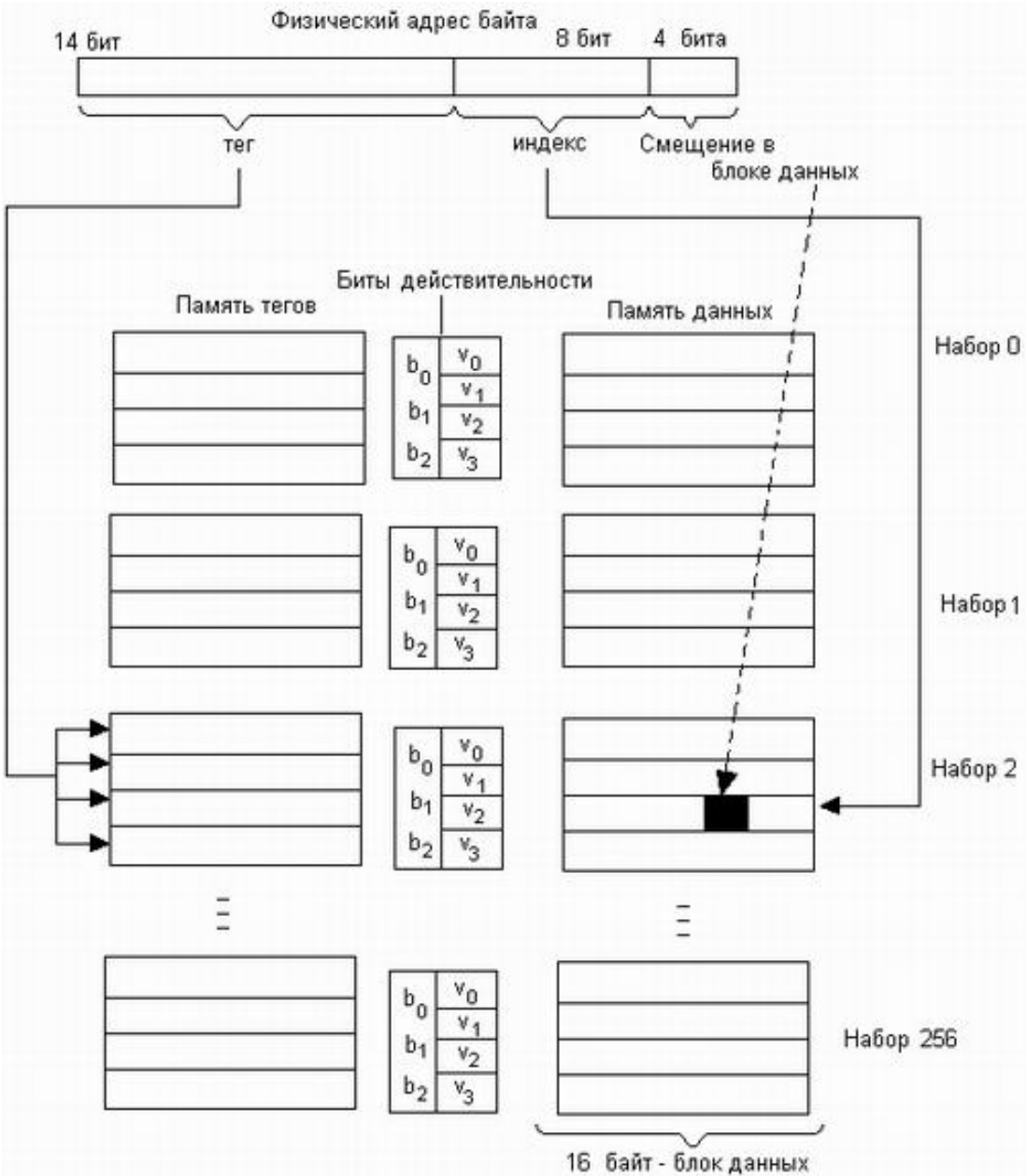
v – бит действительности
b0, b1, b2 – биты обращения



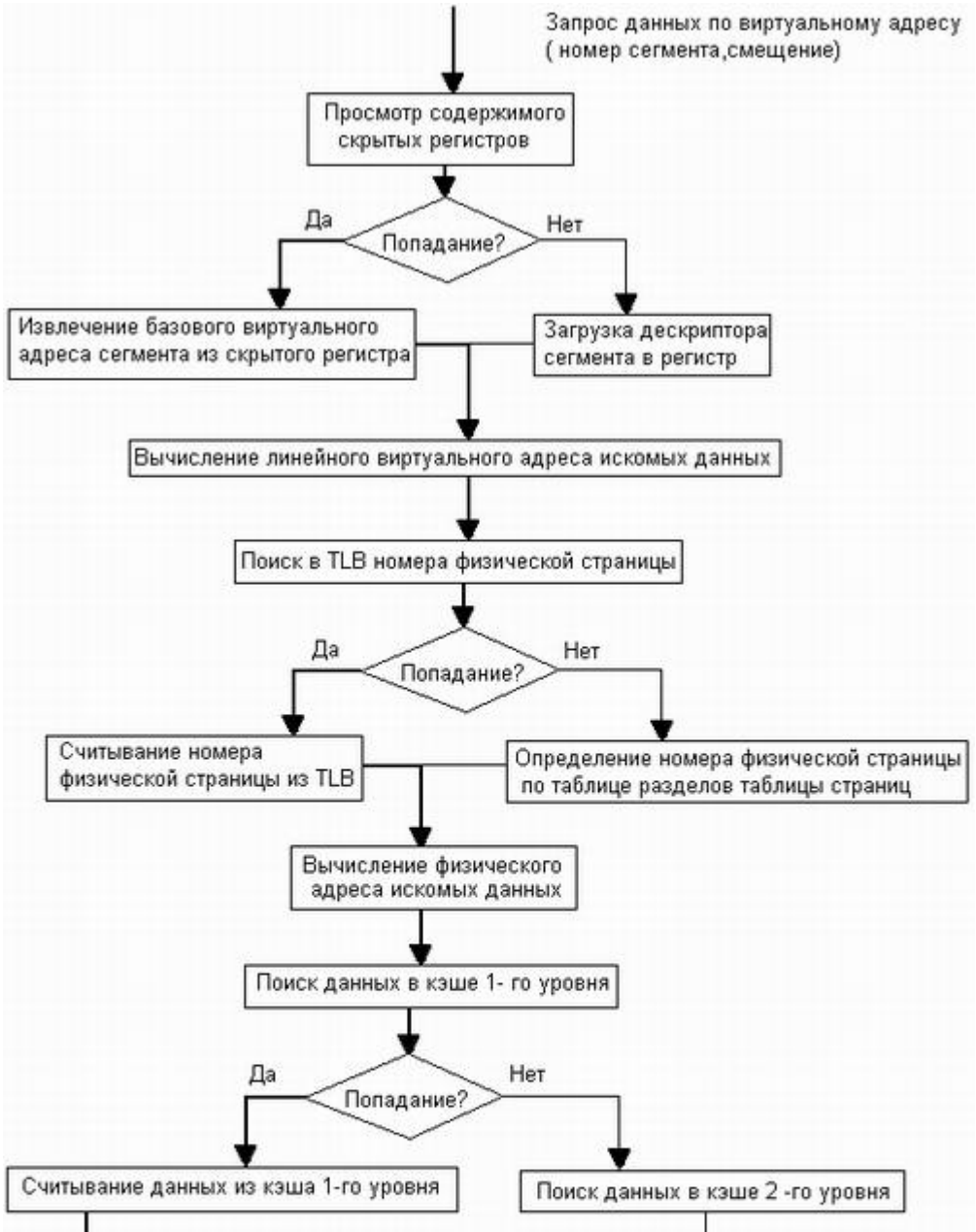
Алгоритм установки битов обращения

L0, если b0=0 и b1=0
L1, если b0=0 и b1=1
L2, если b0=1 и b2=0
L3, если b0=1 и b2=1

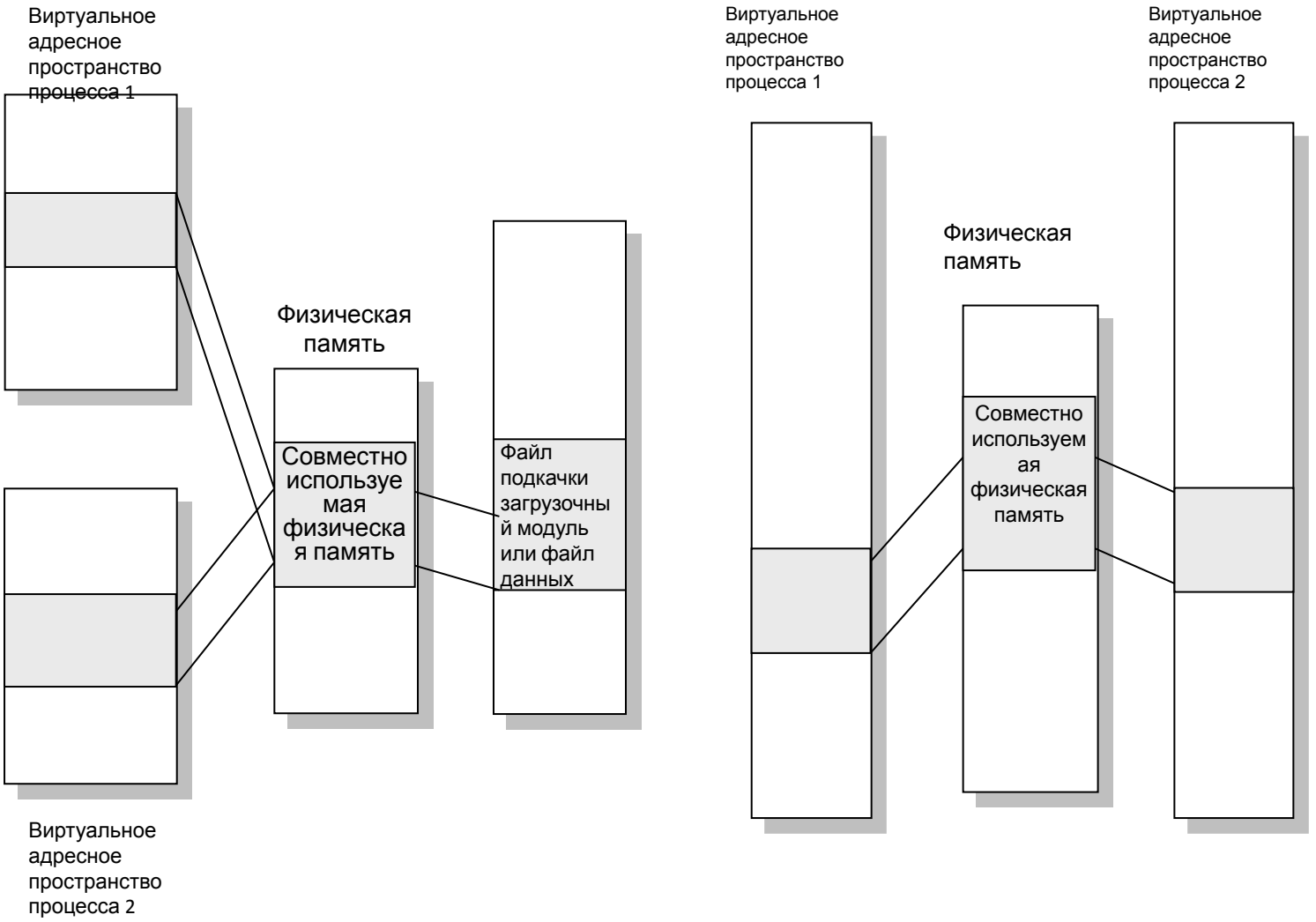
Кэш первого уровня



Совместная работа кэшей разного уровня



Разделяемая память



- Подсистема виртуальной памяти представляет собой удобный механизм для решения задачи совместного доступа нескольких процессов к одному и тому же сегменту памяти, который в этом случае называется разделяемой памятью.

- Для организации разделяемого сегмента достаточно поместить его в виртуальное адресное пространство каждого процесса, которому нужен доступ к данному сегменту, а затем настроить параметры отображения этих виртуальных сегментов так, чтобы они соответствовали одной и той же области оперативной памяти. В этом случае разделяемый сегмент помещается в индивидуальную часть виртуального адресного пространства каждого процесса и описывается в каждом процессе индивидуальным дескриптором сегмента