# Управление памятью

Тема 3

# Функции операционной системы по управлению памятью в мультипрограммных системах

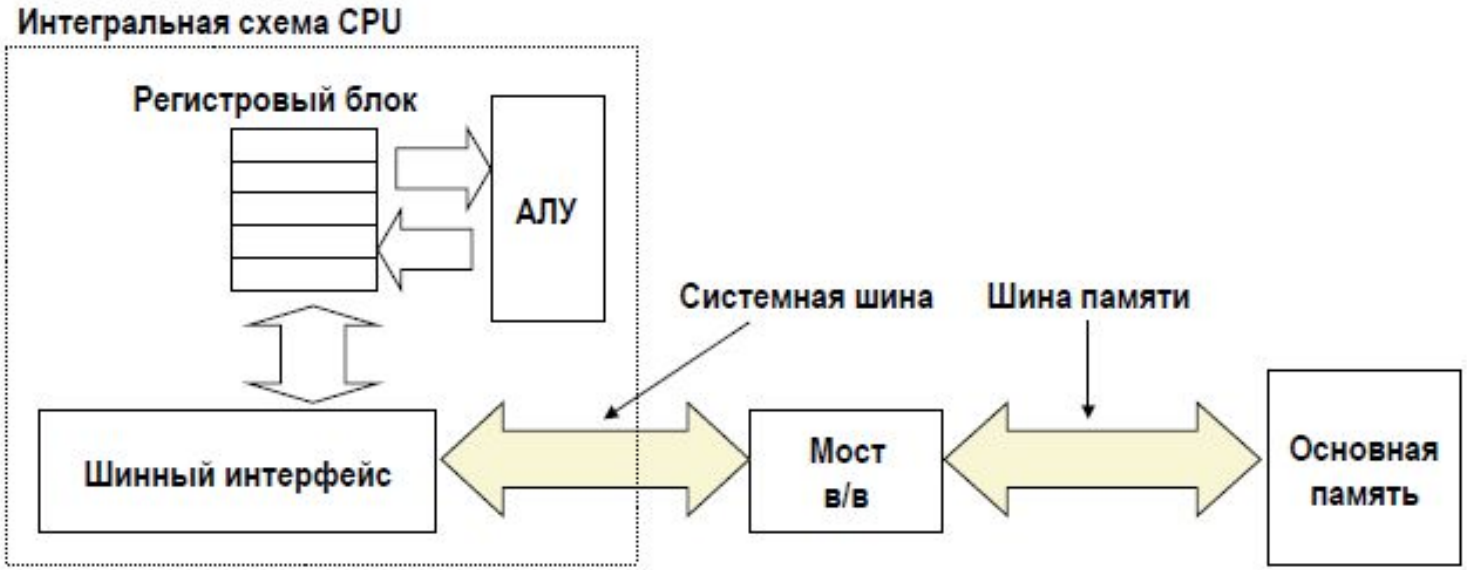
- отслеживание (учет) свободной и занятой памяти;
- первоначальное и динамическое распределение памяти процессов, приложений и самой ОС;
- освобождение памяти при завершении процессов;
- настройка адресов программы на конкретную область физической памяти;
- полное или частичное вытеснение кодов и данных процессов из ОП на диск, когда размеры ОП недостаточны для размещения всех процессов и возвращение их в ОП;
- защита памяти, выделенной процессу, от возможных вмешательств со стороны других процессов;
- дефрагментация памяти.

# Память с произвольным доступом Random-Access Memory (RAM)

- **Ключевые особенности**
  - RAM обычно изготавливается в виде (части) интегральной схемы.
  - Базовая единица хранения - ячейка (1 бит в ячейке).
  - Несколько ИС RAM образуют память.
- **Статическая RAM (SRAM)**
  - Каждая ячейка хранит бит в схеме из 4-6 транзисторов.
  - Под электропитанием сохраняет значение неограниченное время
  - Малочувствительна к электрическим помехам, радиации...
  - Быстрее, крупнее, дороже и прожорливее DRAM.
- **Динамическая RAM (DRAM)**
  - Каждая ячейка хранит бит в схеме из транзистора и конденсатора
  - Содержимое требуется регенерировать каждые 10-100 мс.
  - Чувствительнее чем SRAM к помехам и радиации.
  - Медленнее, мельче, дешевле и экономичнее SRAM.

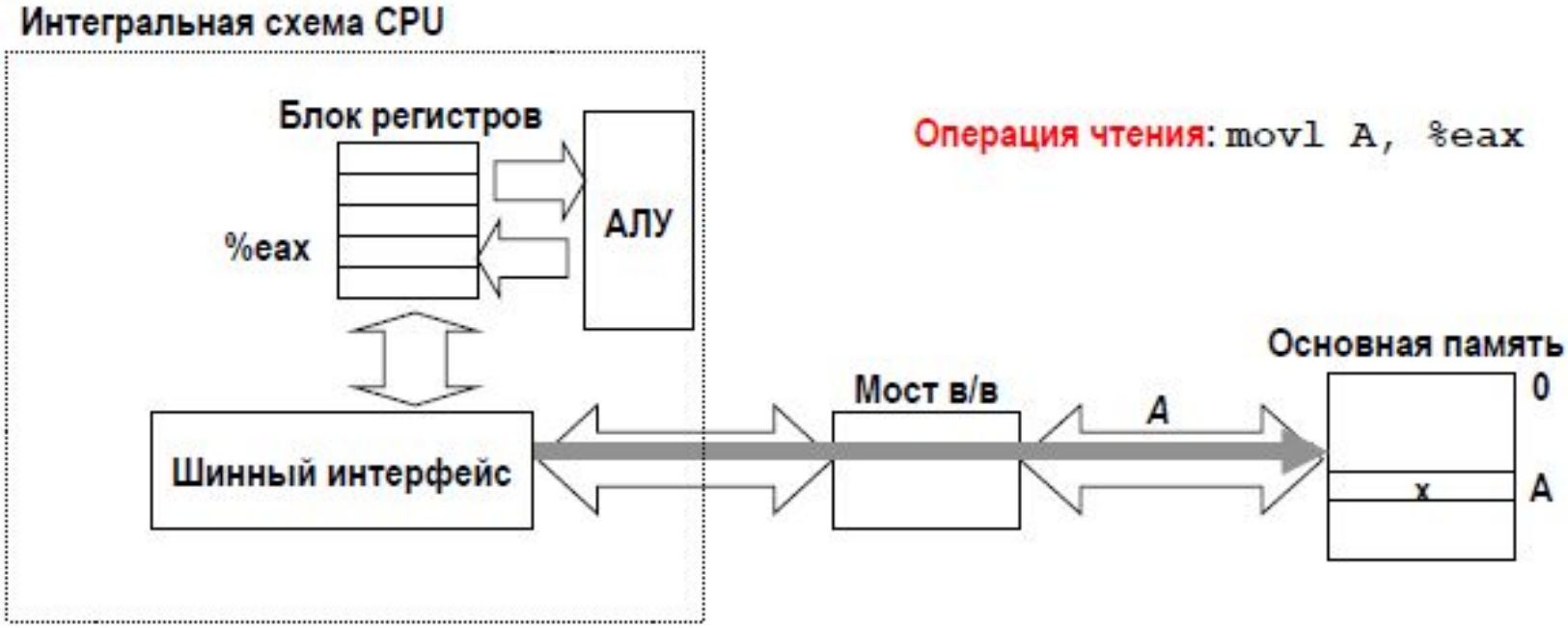
# Традиционное шинное подключение ЦП и основной памяти

- **Шина** набор параллельных проводников, несущих сигналы адреса, данных и управления.
- Обычно к шине подключаются несколько устройств



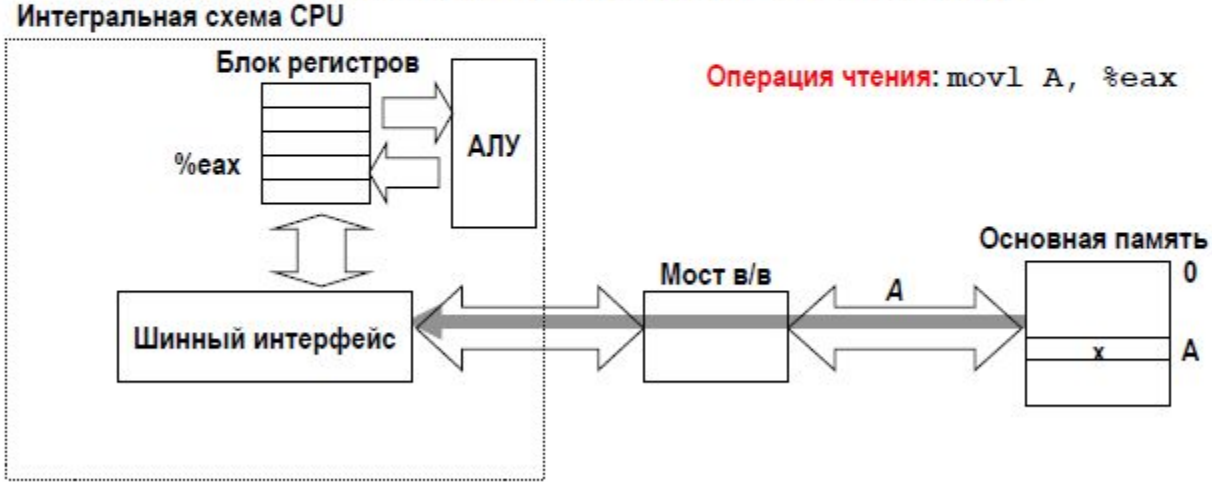
# Чтение из памяти - 1

- ЦП выставляет адрес A на шине памяти



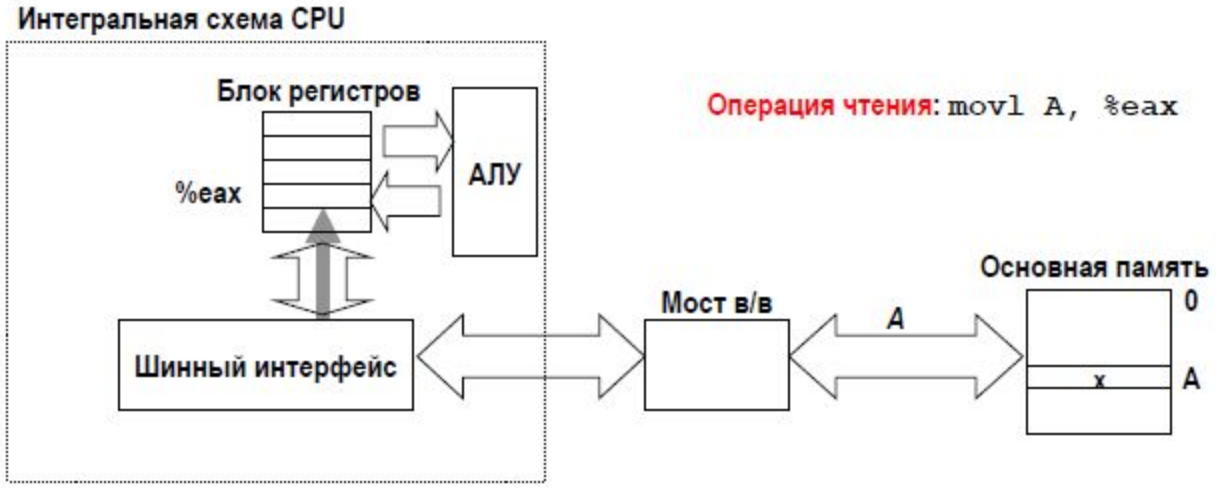
# Чтение из памяти - 2

- Основная память принимает A с шины памяти, выбирает слово x, и выставляет его на шину



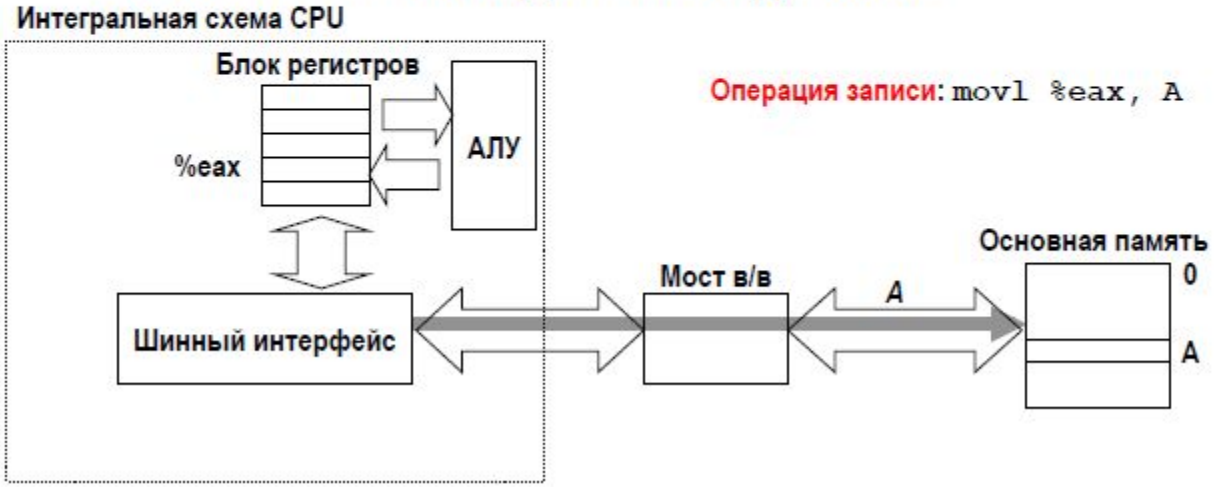
# Чтение памяти - 3

- ЦП считывает слово x с шины и помещает в регистр %eax.



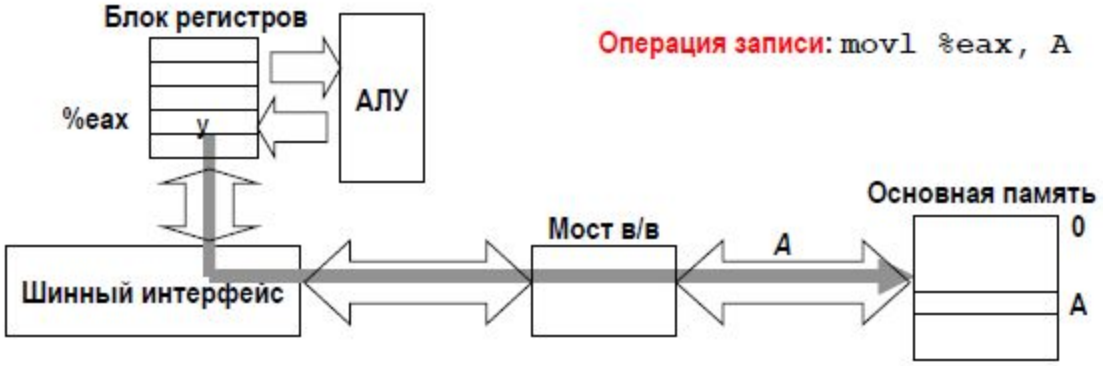
# Запись в память - 1

- ЦП выставляет на шину адрес A. Основная память считывает его и ожидает слова данных.



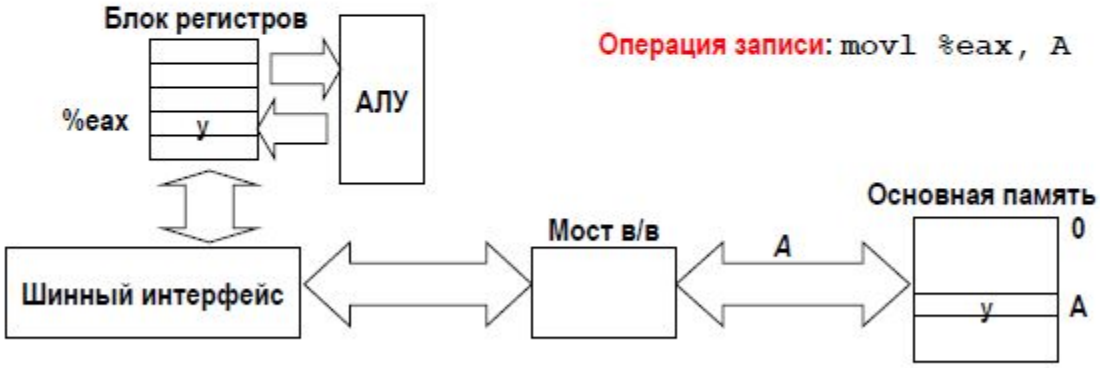
# Запись в память – 2

- ЦП CPU выставляет на шину слово данных y.

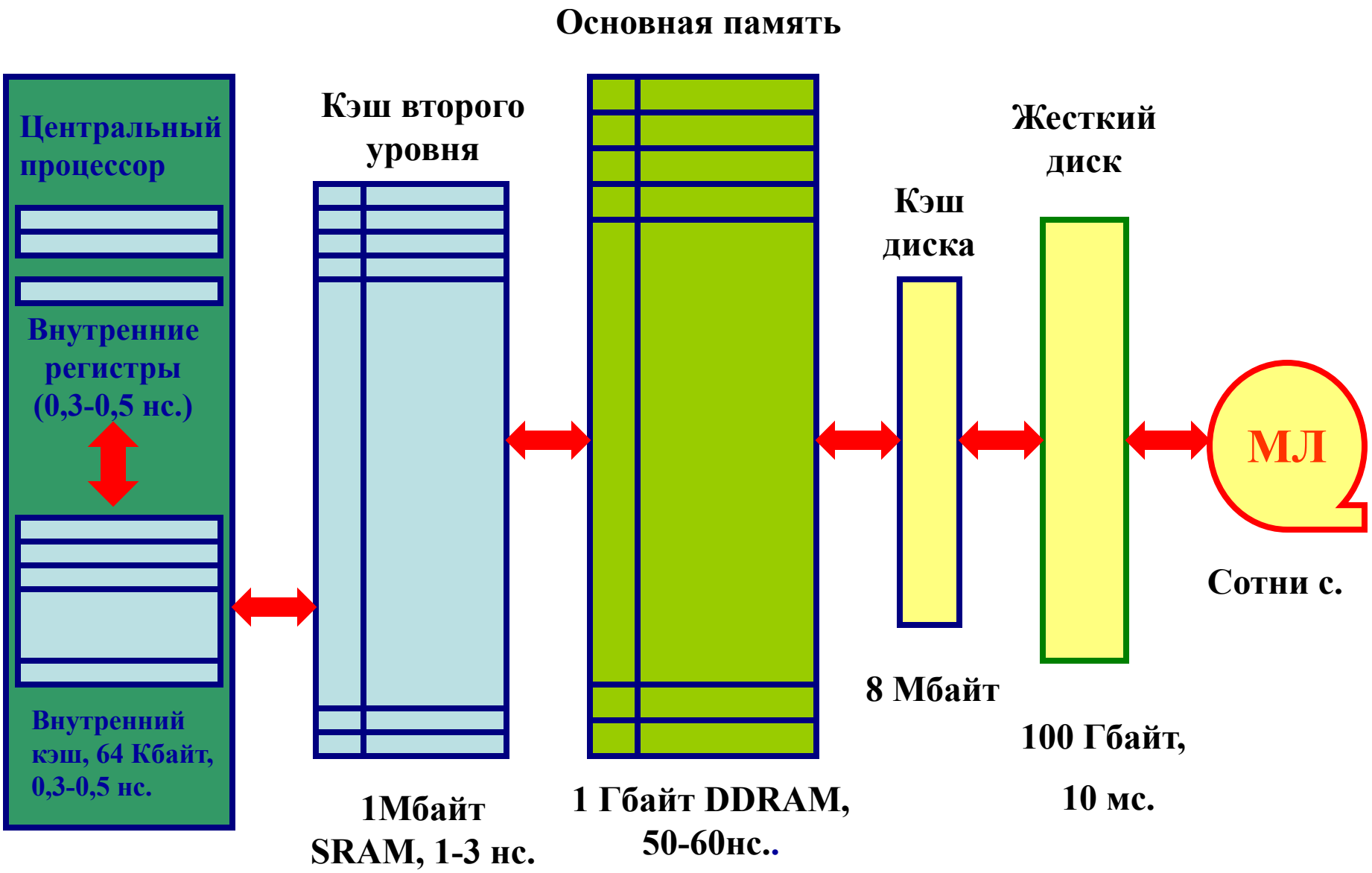


# Запись в память - 3

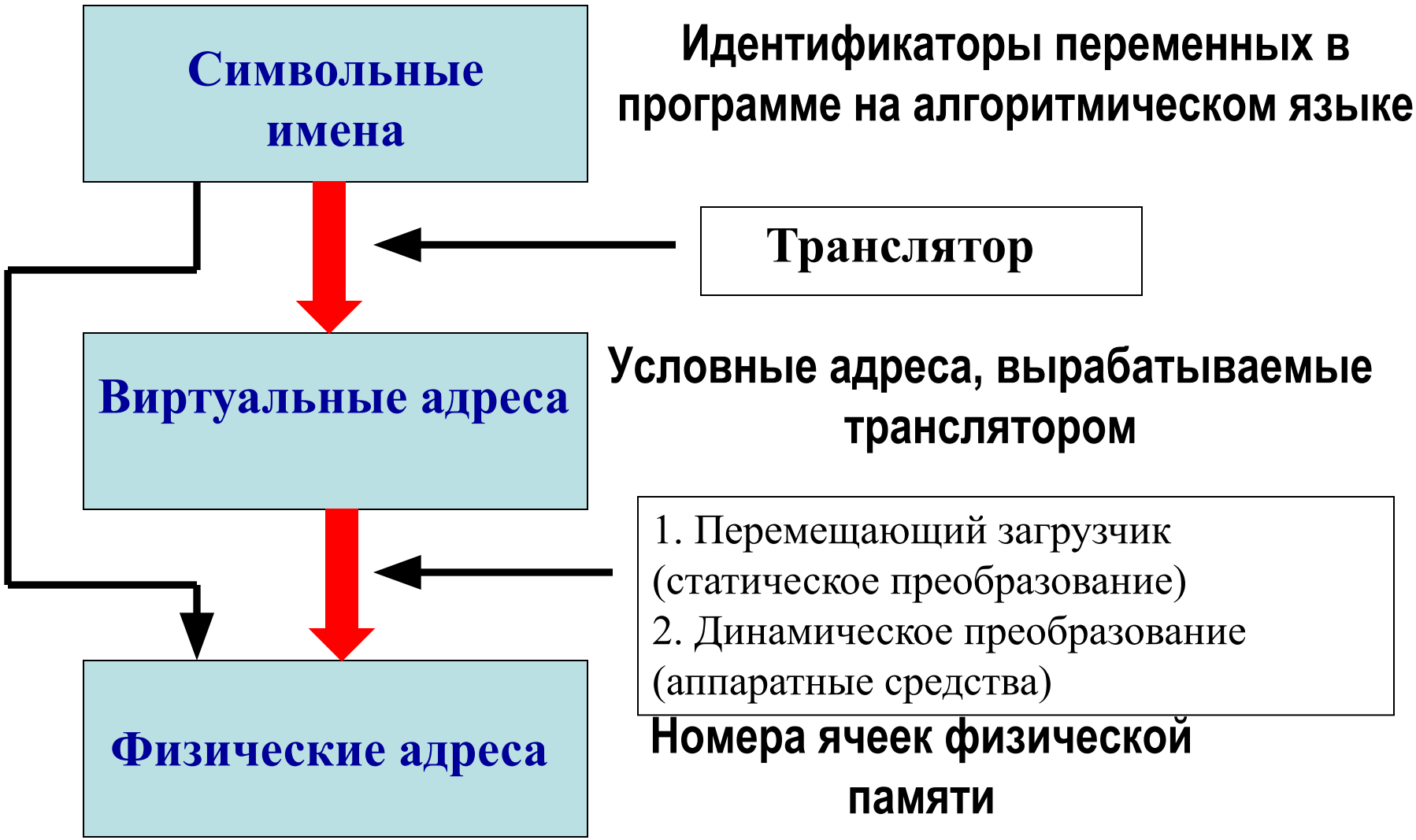
- Память считывает с шины слово данных y и размещает по адресу A.



# Физическая организация памяти

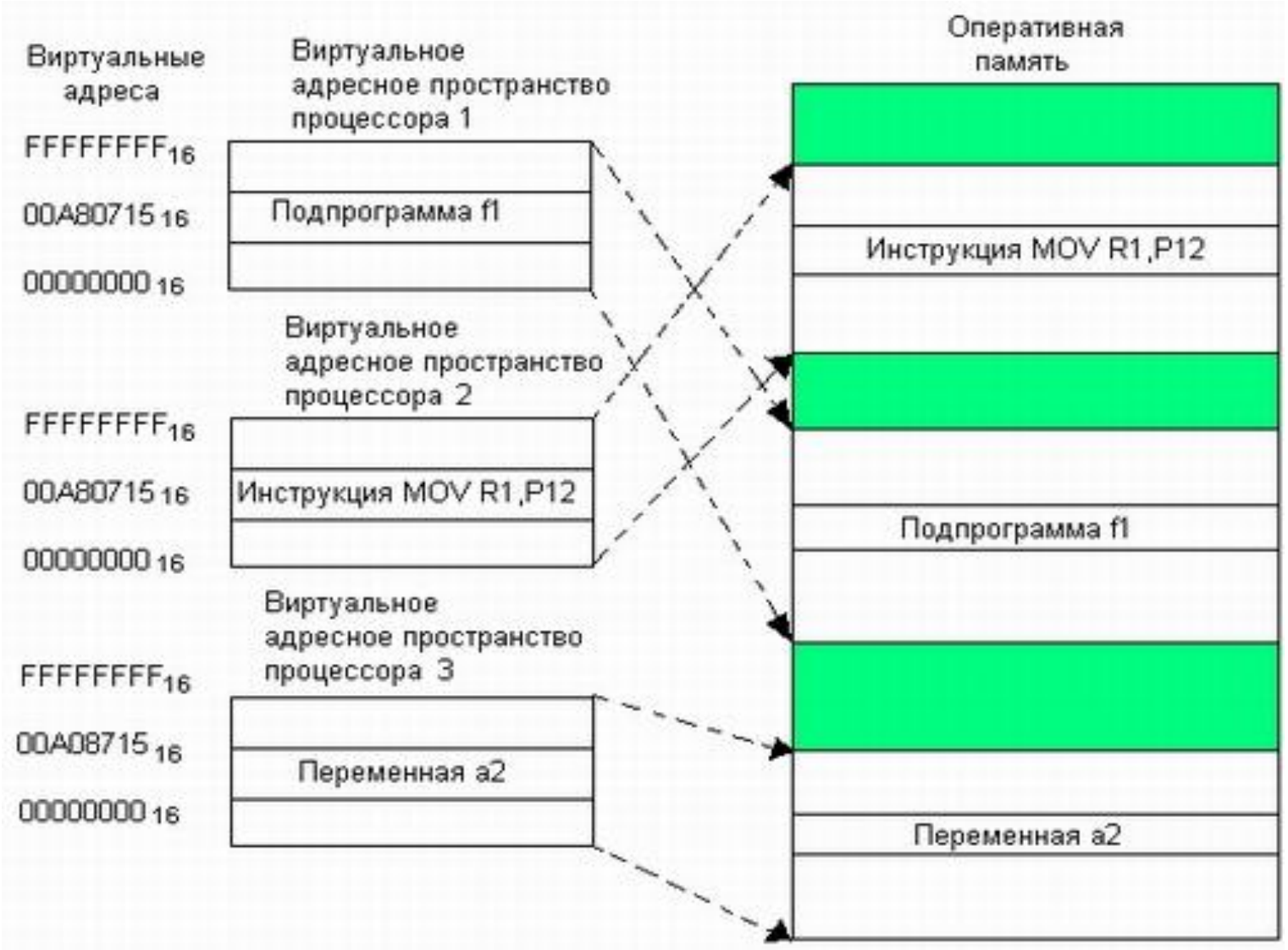


# Типы адресов



Совокупность виртуальных адресов процесса называется **виртуальным адресным пространством.**
Диапазон возможных адресов виртуального пространства у всех процессов является одним и тем же. Например, при использовании 32-разрядных виртуальных адресов этот диапазон задается границами $00000000_{16}$ и $FFFFFFFF_{16}$.
Тем не менее каждый процесс имеет собственное виртуальное адресное пространство — транслятор присваивает виртуальные адреса переменным и кодам каждой программе независимо.
**Максимальный размер виртуального адресного** пространства ограничивается только **разрядностью адреса**, присущей данной архитектуре компьютера, и, как правило, не совпадает с объемом физической памяти, имеющимся в компьютере.

# Виртуальные адресные пространства нескольких программ



# Алгоритмы распределение памяти

## Классификация методов распределения памяти

# Виртуальная память

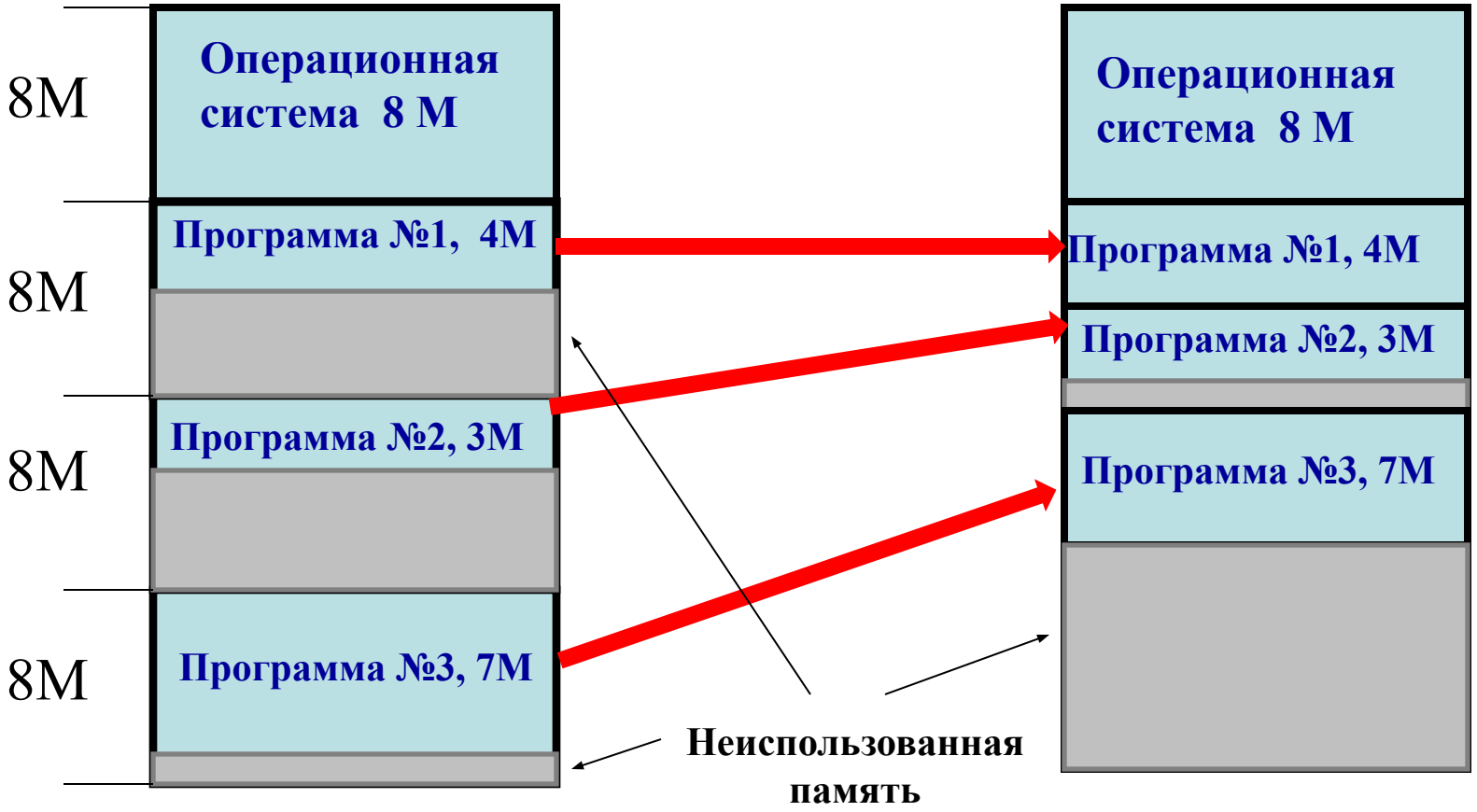
**Методы структуризации виртуального адресного пространства**

**1962 г. – Kilburn T. и др. “One –Level Storage System”**

## Методы реализации виртуальной памяти:

1. **Страничная виртуальная память – организует перемещение данных между ОП и диском страницами – частями виртуального адресного пространства фиксированного и сравнительно небольшого размера.**
2. **Сегментная виртуальная память предусматривает перемещение данных сегментами – частями виртуального адресного пространства произвольного размера, полученными с учетом смыслового значения данных.**
3. **Сегментно-страничная виртуальная память использует двухуровневое деление: виртуальное адресное пространство делится на сегменты, а затем сегменты делятся на страницы. Единицей перемещения данных является страница.**
4. **Для временного хранения сегментов и страниц на диске отводится специальная область – страничный файл или файл подкачки (paging file).**

# Распределение памяти фиксированными разделами (MFT в OS/360)



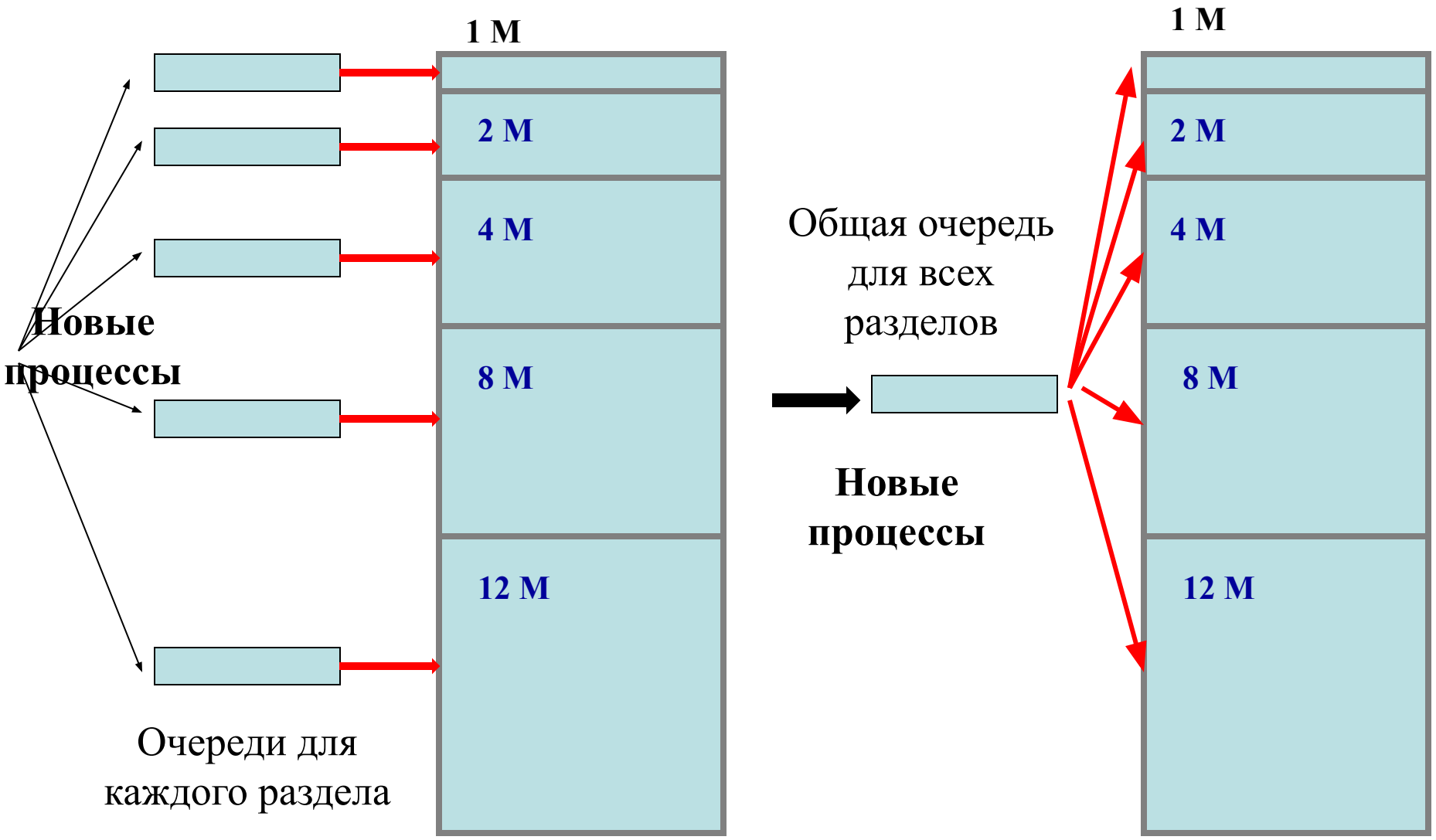

Разделы одинакового размера

Разделы разного размера

# Распределение памяти фиксированными разделами

Подсистема управления памятью в этом случае выполняет следующие задачи:

- Сравнивает объем памяти, требуемый для вновь поступившего процесса, с размерами свободных разделов и выбирает подходящий раздел;
- Осуществляет загрузку программы в один из разделов и настройку адресов. Уже на этапе трансляции разработчик программы может задать раздел, в котором ее следует выполнять. Это позволяет сразу, без использования перемещающего загрузчика, получить машинный код, настроенный на конкретную область памяти.


1 M
2 M
4 M
8 M
12 M
Новые процессы
Очереди для каждого раздела
Общая очередь для всех разделов
Новые процессы
1 M
2 M
4 M
8 M
12 M

# Распределение памяти фиксированными разделами

**1. Разделы одинакового размера. Недостатки:**

- **необходимость разработки оверлеев при больших размерах программ;**
- **неэффективное использование памяти (внутренняя фрагментация)**

**2. Разделы разного размера. Очередь к каждому разделу.**

**Достоинство: возможность распределения процессов между разделами с минимизацией внутренней фрагментации.**

**Недостаток: возможно неэффективное использование памяти за счет «простоя» больших разделов при наличии только небольших процессов.**

**3. Разделы разного размера. Общая очередь к разделам.**

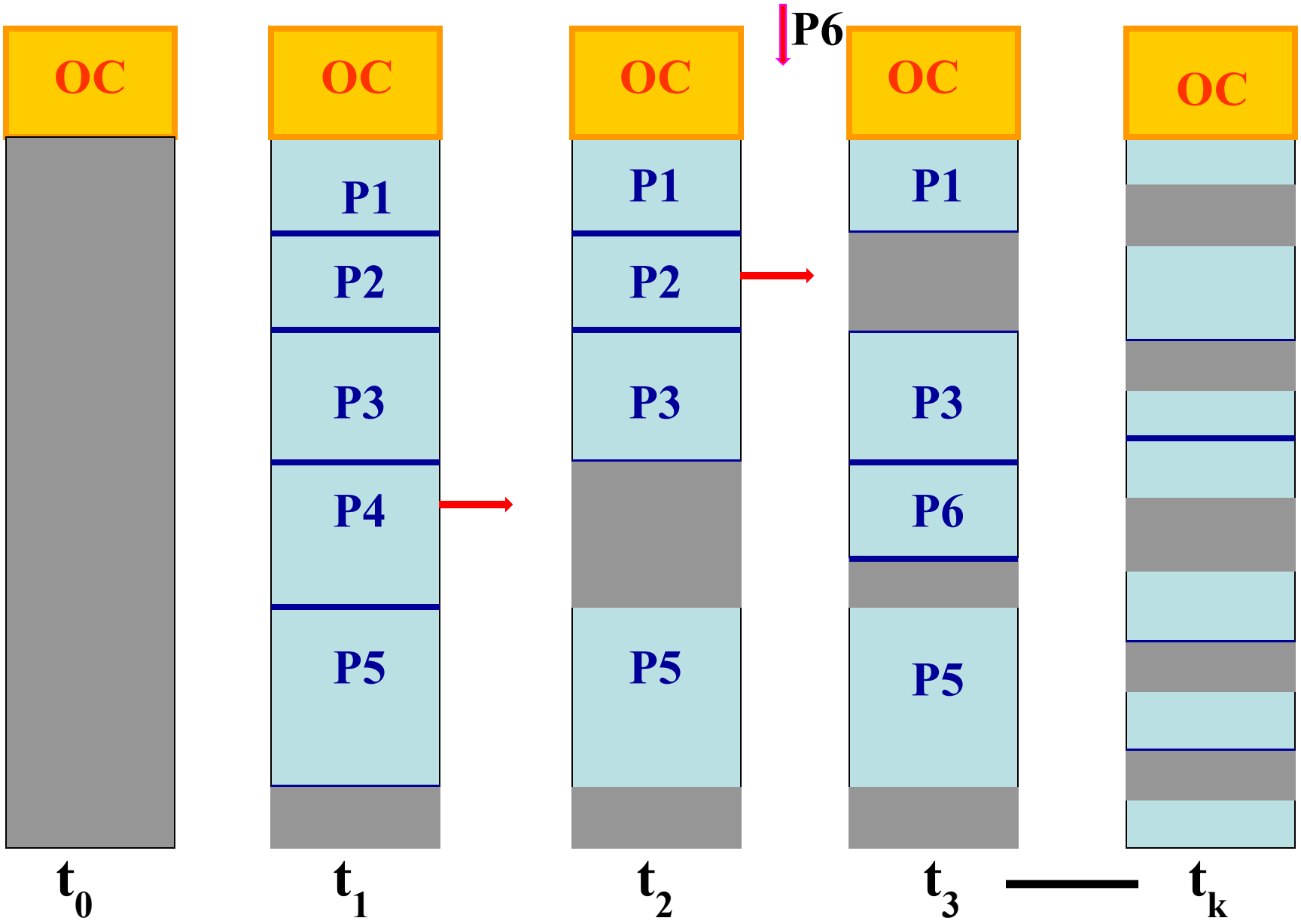
**Достоинство: улучшается использование памяти.**

**Достоинства: простота, минимальные требования к операционной системе. Недостатки: 1) количество разделов, определенных во время генерации ОС (режим MFT OS/360), ограничивает число активных процессов; 2) неэффективное использование памяти.**

# Распределение памяти динамическими разделами

В этом случае память машины не делится заранее на разделы. Сначала вся память, отводимая для приложений, свободна. Каждому вновь поступающему на выполнение приложению на этапе создания процесса выделяется вся необходимая ему память (если достаточный объем памяти отсутствует, то приложение не принимается на выполнение и процесс для него не создается). После завершения процесса память освобождается, и на это место может быть загружен другой процесс. Таким образом, в произвольный момент времени оперативная память представляет собой случайную последовательность занятых и свободных участков (разделов) произвольного размера.

# Распределение памяти динамическими разделами



# Распределение памяти динамическими разделами. Функции ОС

- Ведение таблиц свободных и занятых областей, в которых указываются начальные адреса и размеры участков памяти.
- При создании нового процесса — анализ требований к памяти, просмотр таблицы свободных областей и выбор раздела, размер которого достаточен для размещения кодов и данных нового процесса.
- Загрузка программы в выделенный ей раздел и корректировка таблиц свободных и занятых областей. Данный способ предполагает, что программный код не перемещается во время выполнения, а значит, настройка адресов может быть проведена единовременно во время загрузки.
- После завершения процесса корректировка таблиц свободных и занятых областей

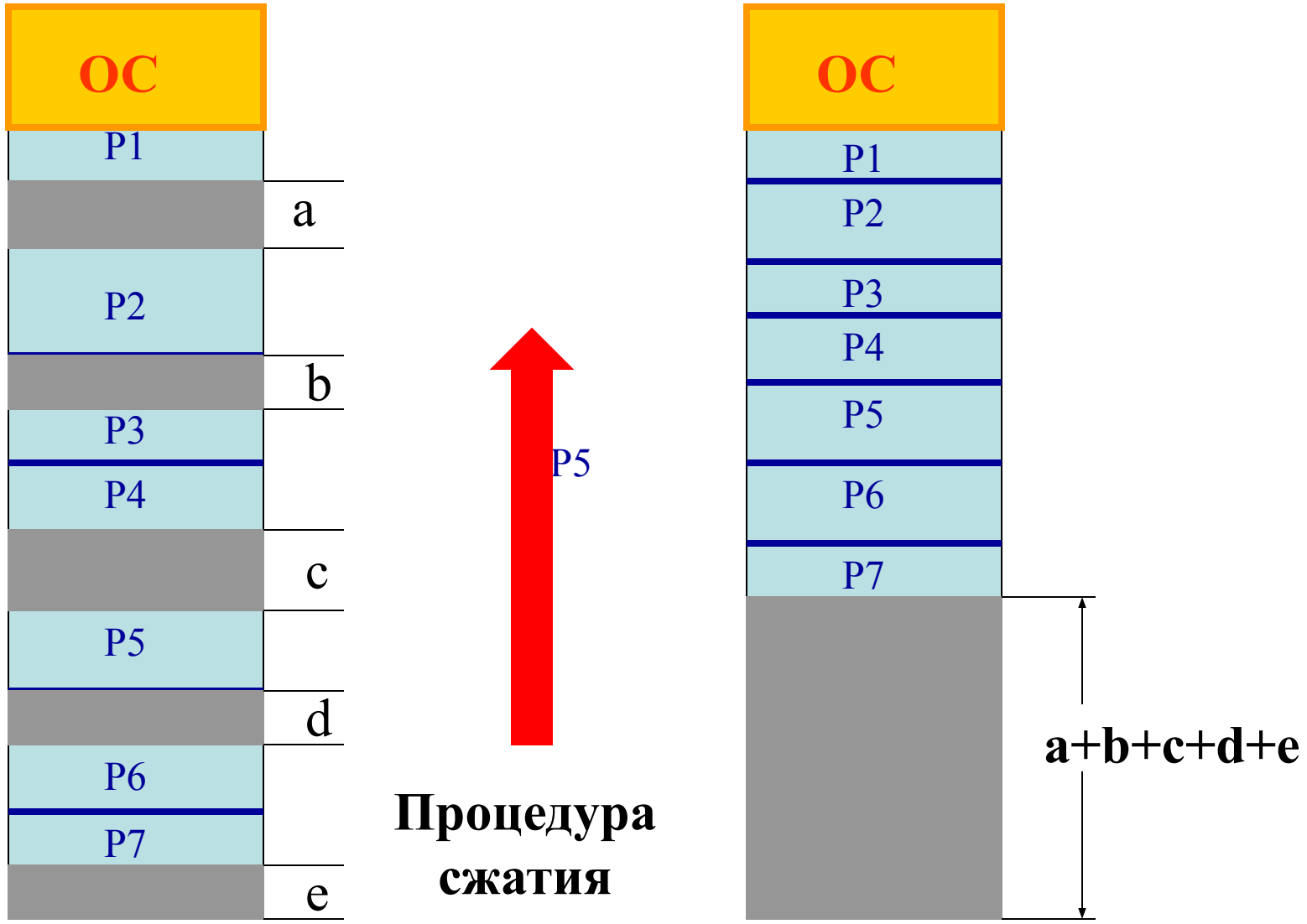
# Распределение памяти динамическими разделами

**<u>Достоинства:</u> большая гибкость по сравнению с фиксированными разделами. <u>Недостаток:</u> внешняя фрагментация**

**Функции ОС для реализации метода MVT OS/360 (ЕС ЭВМ):**

- **ведение таблиц свободных и занятых областей ОП с указанием начального адреса и размера ;**
- **при создании нового раздела просмотр таблиц и выбор раздела, достаточного для размещения процесса (наименьший или наибольший достаточный из свободных);**
- **загрузка процесса в выделенный раздел и корректировка таблиц свободных и занятых областей основной памяти;**
- **после завершения процесса корректировка таблиц свободных и занятых областей.**

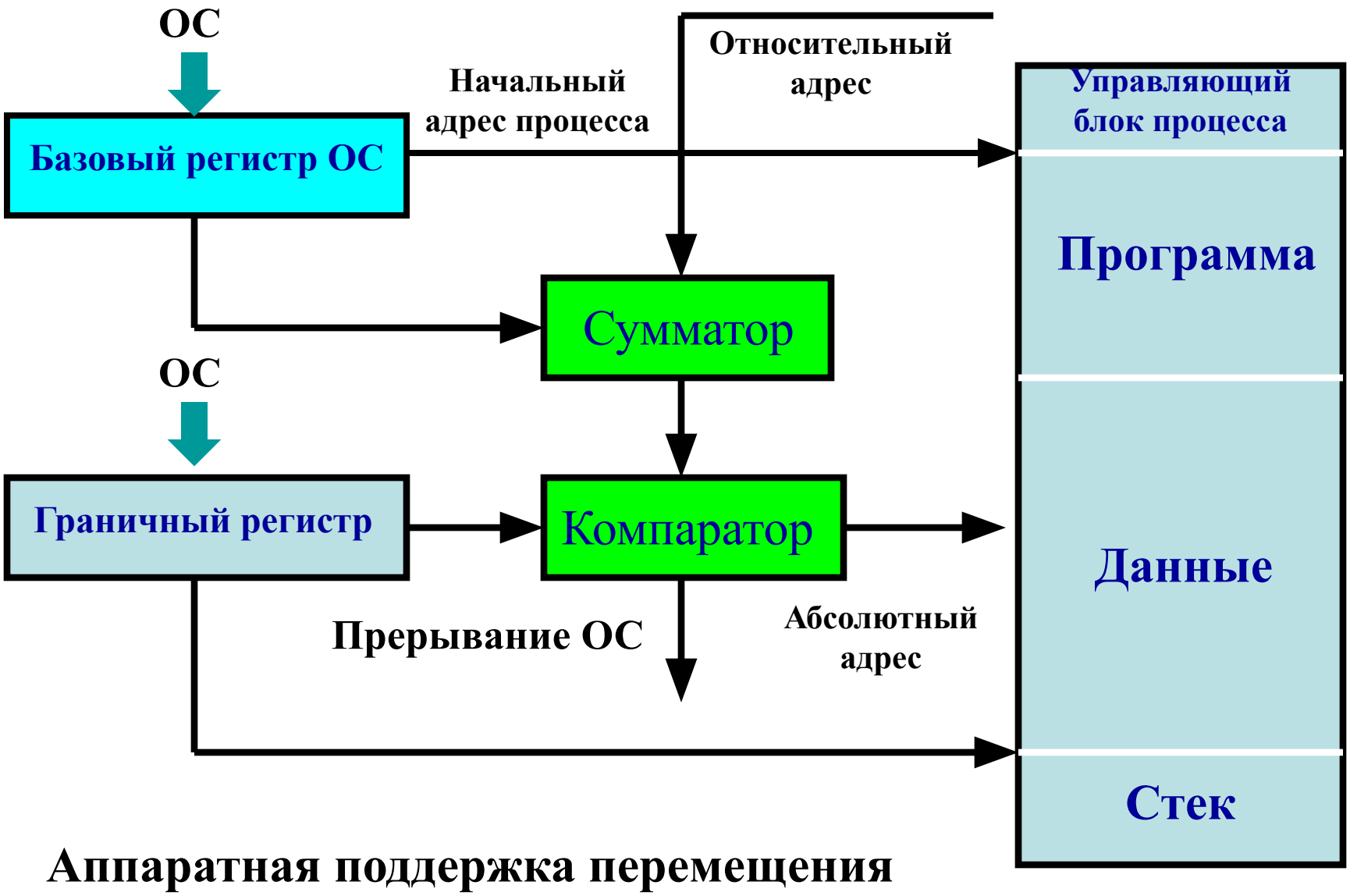
# Распределение памяти перемещаемыми разделами



# Распределение памяти перемещаемыми разделами

1. **Перемещение всех занятых участков в сторону старших или младших адресов при каждом завершении процесса или для вновь создаваемого процесса в случае отсутствия раздела достаточного размера.**
2. **Коррекция таблиц свободных и занятых областей.**
3. **Изменение адресов команд и данных, к которым обращаются процессы при их перемещении в памяти за счет использования относительной адресации.**
4. **Аппаратная поддержка процесса динамического преобразования относительных адресов в абсолютные адреса основной памяти.**
5. **Защита памяти, выделяемой процессу, от взаимного влияния других процессов.**

Достоинства распределения памяти перемещаемыми разделами: **эффективное использование оперативной памяти, исключение внутренней и внешней фрагментации.** Недостаток: **дополнительные накладные расходы ОС.**




**Аппаратная поддержка перемещения**

C:\WINDOWS\system32\cmd.exe

```
C:\>mem /p

  Адрес        Имя          Размер       Тип
  ------       --------     ------       ------
  000000                    000400       Вектор прерывания
  000400                    000100       Область обмена ПЗУ (ROM)
  000500                    000200       Область обмена DOS

  000700       IO           000370       Системные данные

  000A70       MSDOS        0016A0       Системные данные

  002110       IO           002230       Системные данные
                 KBD        000CE0        Системная программа
                 HIMEM      0004E0        DEVICE=
                 HASPDOS    000150        DEVICE=
                            000490        FILES=
                            000090        FCBS=
                            0001B0        LASTDRIVE=
                            0007E0        STACKS=
  004350       COMMAND      000B50       Программа
  004EB0       MSDOS        000070       - Свободно -
  004F30       COMMAND      0004D0       Окружение
  005410       MEM          000410       Окружение
  005830       MEM          0174E0       Программа
  01CD20       MSDOS        0832C0       - Свободно -
  09FFF0       SYSTEM       02E000       Системная программа

  0CE000       IO           003100       Системные данные
                 MOUSE      0030F0        Системная программа
  0D1110       MSDOS        000400       - Свободно -
  0D1520       MSCDEXNT     0001D0       Программа
  0D1700       REDIR        000A70       Программа
  0D2180       DOSX         0087A0       Программа
  0DA930       DOSX         000080       Данные
  0DA9C0       MSDOS        005630       - Свободно -


    655360 байт - всего обычной памяти
    655360 байт - доступно для MS-DOS
    632752 максимальный размер исполняемой программы

   1048576 байт - всего непрерывной дополнительной памяти
         0 байт - доступно непрерывной дополнительной памяти
    941056 байт - доступной памяти XMS
           резидентная часть MS-DOS загружена в сегмент HMA
```

# Виртуальная память

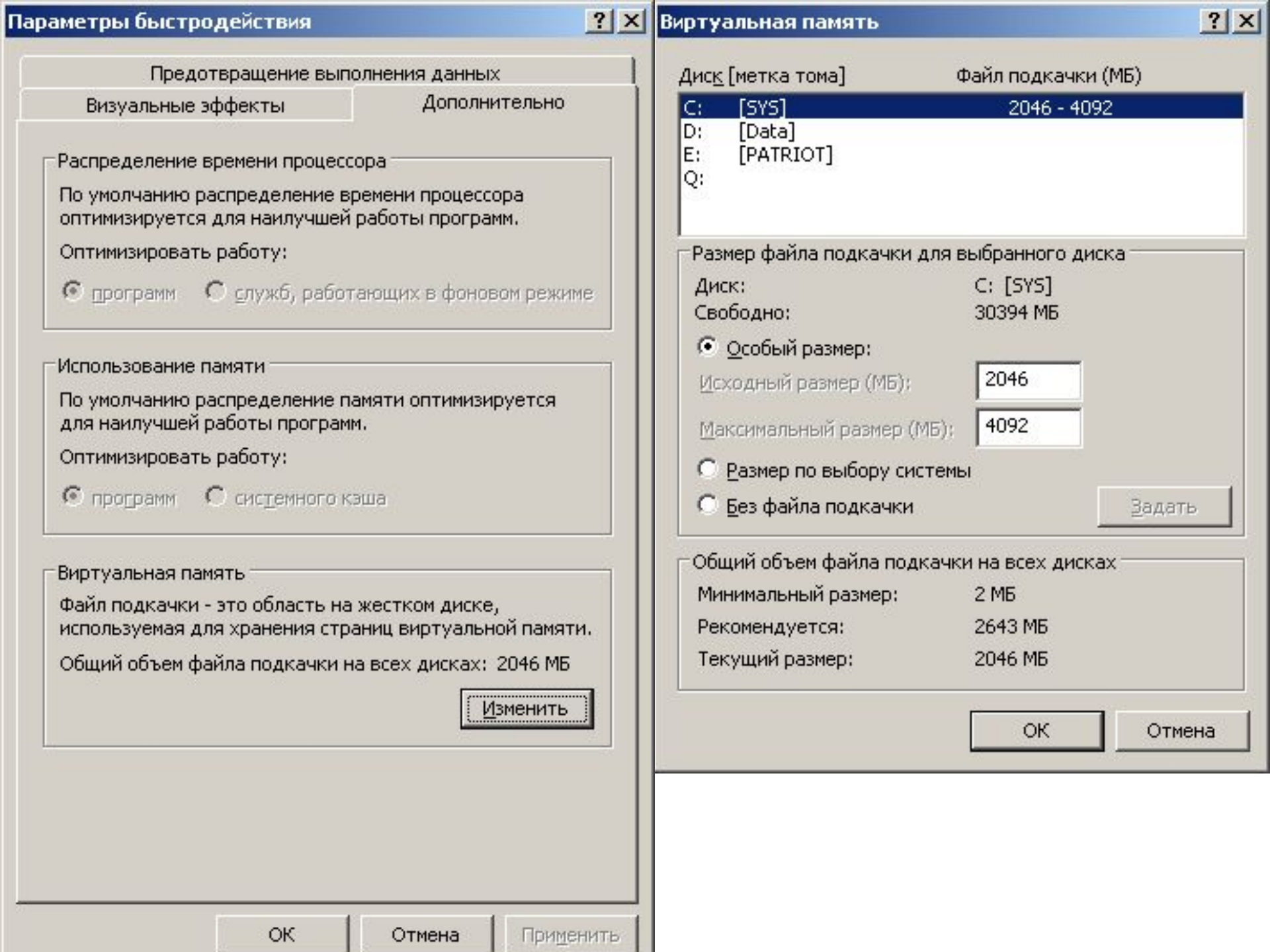
Виртуальным называется ресурс, который пользователю или пользовательской программе представляется обладающим свойствами, которыми он в действительности не обладает.

В данном случае в распоряжение прикладного программиста предоставляется виртуальная оперативная память, размер которой намного превосходит всю имеющуюся в системе реальную оперативную память. Виртуализация памяти может быть осуществлена на основе двух различных подходов:

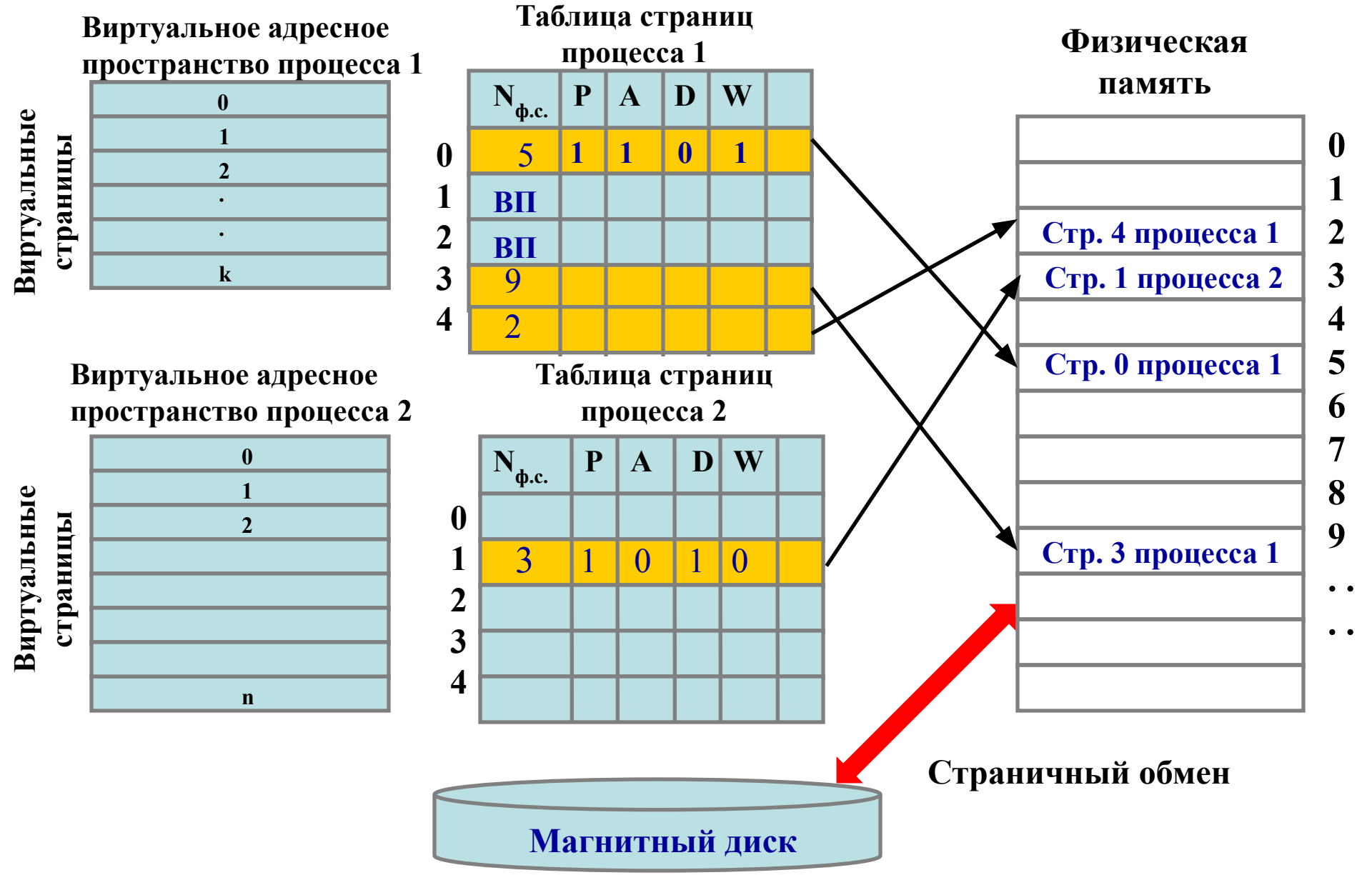
- **свопинг** (swapping) — образы процессов выгружаются на диск и возвращаются в оперативную память целиком;
- **виртуальная память** (virtual memory) — между оперативной памятью и диском перемещаются части (сегменты, страницы и т. п.) образов процессов.

Виртуализация ОП осуществляется совокупностью программных модулей ОС и аппаратных схем процессора и включает решение следующих задач:

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


Параметры быстродействия
Предотвращение выполнения данных
Визуальные эффекты
Дополнительно
Распределение времени процессора
По умолчанию распределение времени процессора оптимизируется для наилучшей работы программ.
Оптимизировать работу:
программ
служб, работающих в фоновом режиме
Использование памяти
По умолчанию распределение памяти оптимизируется для наилучшей работы программ.
Оптимизировать работу:
программ
системного кэша
Виртуальная память
Файл подкачки - это область на жестком диске, используемая для хранения страниц виртуальной памяти.
Общий объем файла подкачки на всех дисках: 2046 МБ
Изменить
OK
Отмена
Применить
Виртуальная память
Диск [метка тома]
Файл подкачки (МБ)
C: [SYS] 2046 - 4092
D: [Data]
E: [PATRIOT]
Q:
Размер файла подкачки для выбранного диска
Диск: C: [SYS]
Свободно: 30394 МБ
Особый размер:
Исходный размер (МБ): 2046
Максимальный размер (МБ): 4092
Размер по выбору системы
Без файла подкачки
Задать
Общий объем файла подкачки на всех дисках
Минимальный размер: 2 МБ
Рекомендуется: 2643 МБ
Текущий размер: 2046 МБ
OK
Отмена

# Страничная организация виртуальной памяти



Запись таблицы, называемая дескриптором страницы, включает следующую информацию:

- номер физической страницы, в которую загружена данная виртуальная страница;
- признак присутствия, устанавливаемый в единицу, если виртуальная страница находится в оперативной памяти;
- признак модификации страницы, который устанавливается в единицу всякий раз, когда производится запись по адресу, относящемуся к данной странице;
- признак обращения к странице, называемый также битом доступа, который устанавливается в единицу при каждом обращении по адресу, относящемуся к данной странице.

Виртуальное адресное пространство каждого процесса делится на части одинакового, фиксированного для данной системы размера, называемые виртуальными страницами (virtual pages). В общем случае размер виртуального адресного пространства процесса не кратен размеру страницы, поэтому последняя страница каждого процесса дополняется фиктивной областью.
Вся оперативная память машины также делится на части такого же размера, называемые физическими страницами (или блоками, или кадрами). Размер страницы выбирается равным степени двойки: 512, 1024, 4096 байт и т. д.

Двоичное представление адресов:

**00**0 000 000 000

000 000 000 001

000 000 000 010

- 
- 
- 
- 

номер нулевой страницы

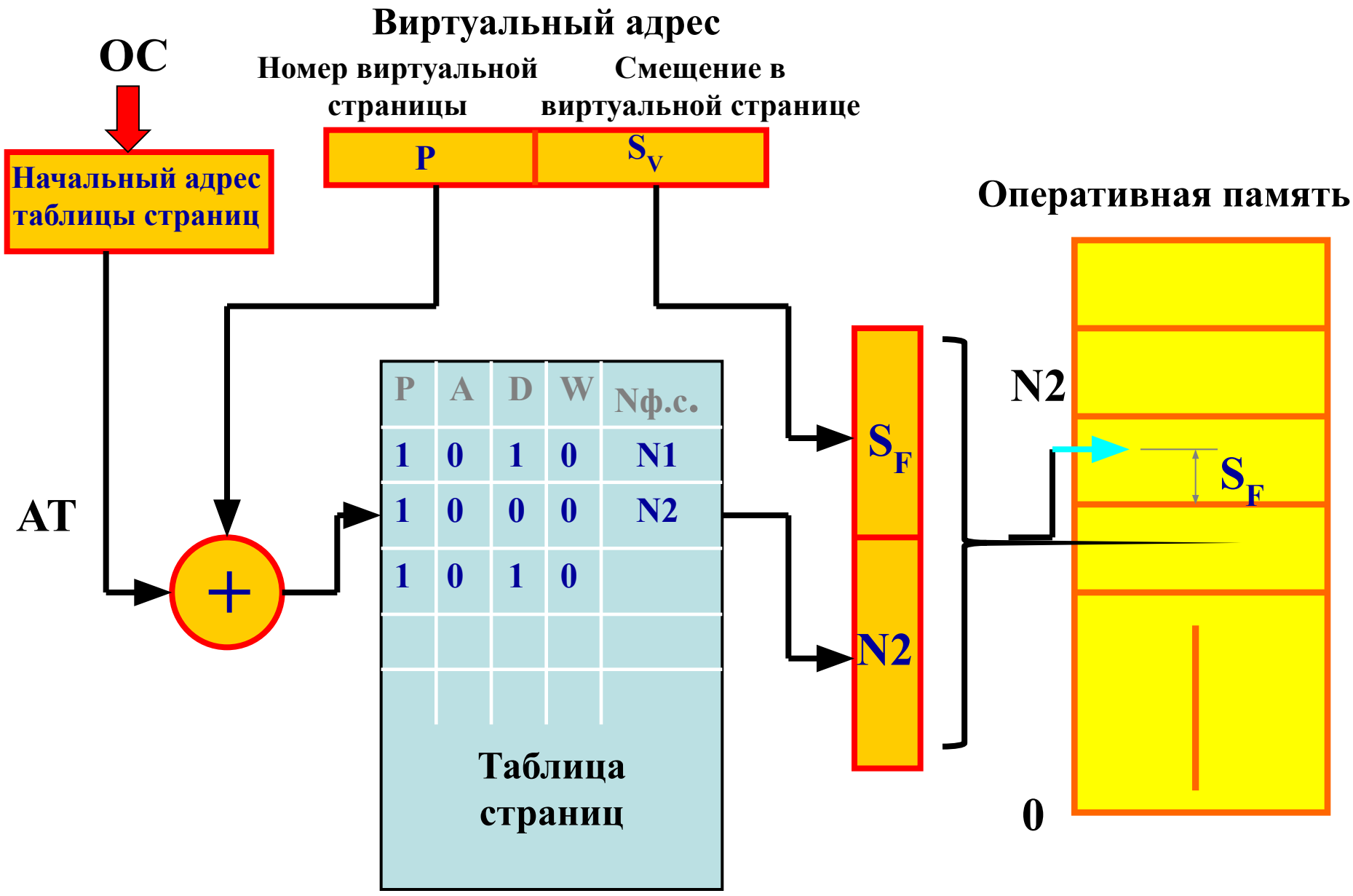
**00**1 111 111 111 – конец нулевой страницы

**01**0 000 000 000 – начало первой страницы

010 000 000 010

- 
- 
- 
- 

**01**1 111 111 111 – конец первой страницы


ОС
Начальный адрес таблицы страниц
Виртуальный адрес
Номер виртуальной страницы
Смещение в виртуальной странице
P
$S_V$
Оперативная память
AT
+
P A D W Nф.с.
1 0 1 0 N1
1 0 0 0 N2
1 0 1 0
Таблица страниц
$S_F$
N2
N2
$S_F$
0

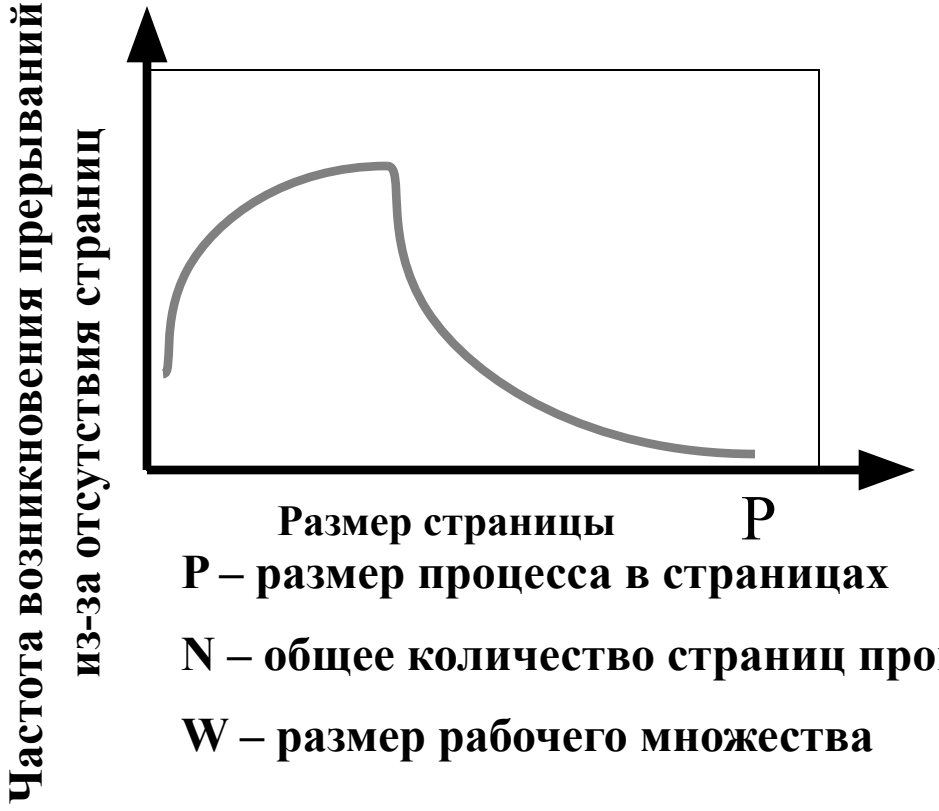
# Оптимизация функционирования страничной виртуальной памяти

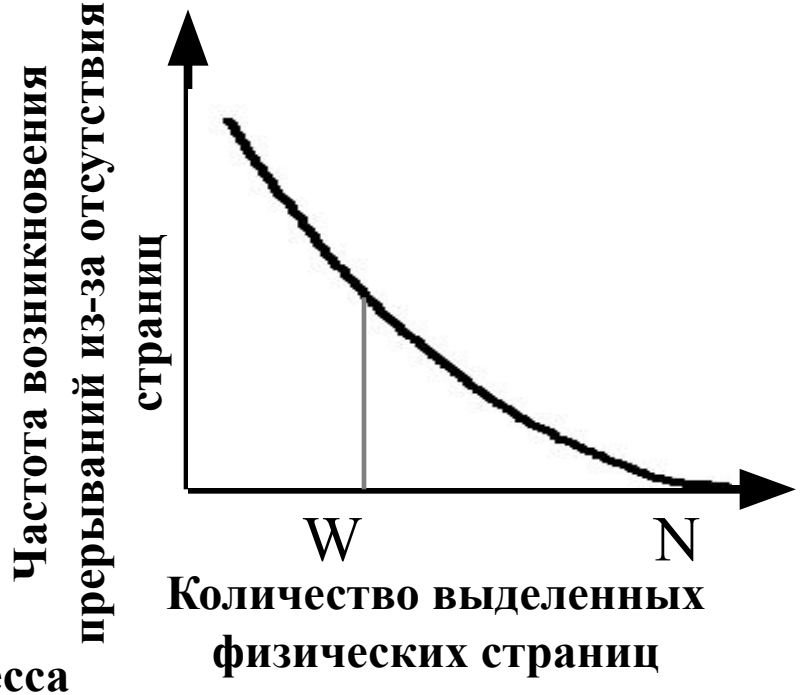
## Методы повышения эффективности функционирования страничной виртуальной памяти:

1. Структуризация виртуального адресного пространства, например, двухуровневая (типичная для 32-битовой адресации).
2. Хранение активной части записей таблицы страниц в высокоскоростном КЭШе или буфере быстрого преобразования адреса (translation lookaside buffer – TLB).
3. Выбор оптимального размера страниц.
4. Эффективное управление страничным обменом, использование оптимальных алгоритмов замены страниц.

# Оптимальный размер страниц

1. С уменьшением размера страницы уменьшается внутренняя фрагментация.
2. С уменьшением размера страницы увеличивается объем страничных таблиц и следовательно накладные расходы на работу виртуальной памяти.
3. С увеличением размера страниц повышается скорость работы диска.
4. Частота страничных прерываний нелинейно зависит от размера страниц




P – размер процесса в страницах

N – общее количество страниц процесса

W – размер рабочего множества

# Управление страничным обменом

Задачи управления страничным обменом:

**- когда передавать страницу в основную память;**

**- где размещать страницу в физической памяти;**

**- какую страницу основной памяти выбирать для замещения, если в основной памяти нет свободных страниц;**

**- сколько страниц процесса следует загрузить в основную память;**

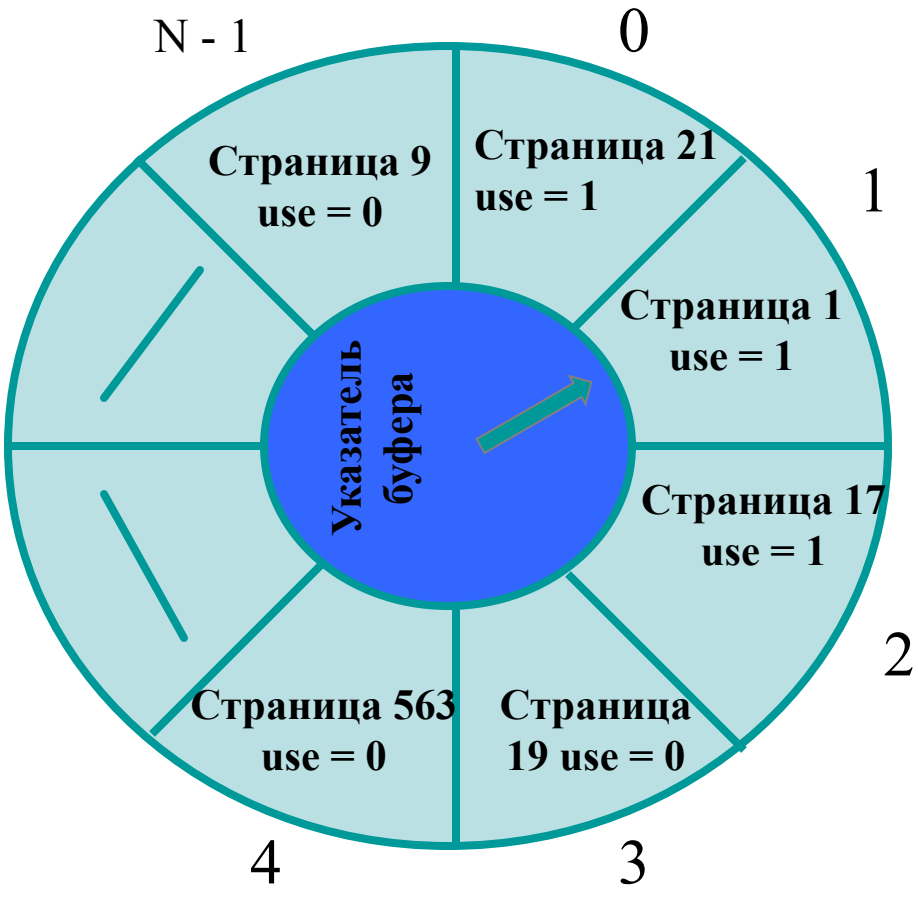
**- когда измененная страница должна быть записана во вторичную память;**

**- сколько процессов размещать в основной памяти.**

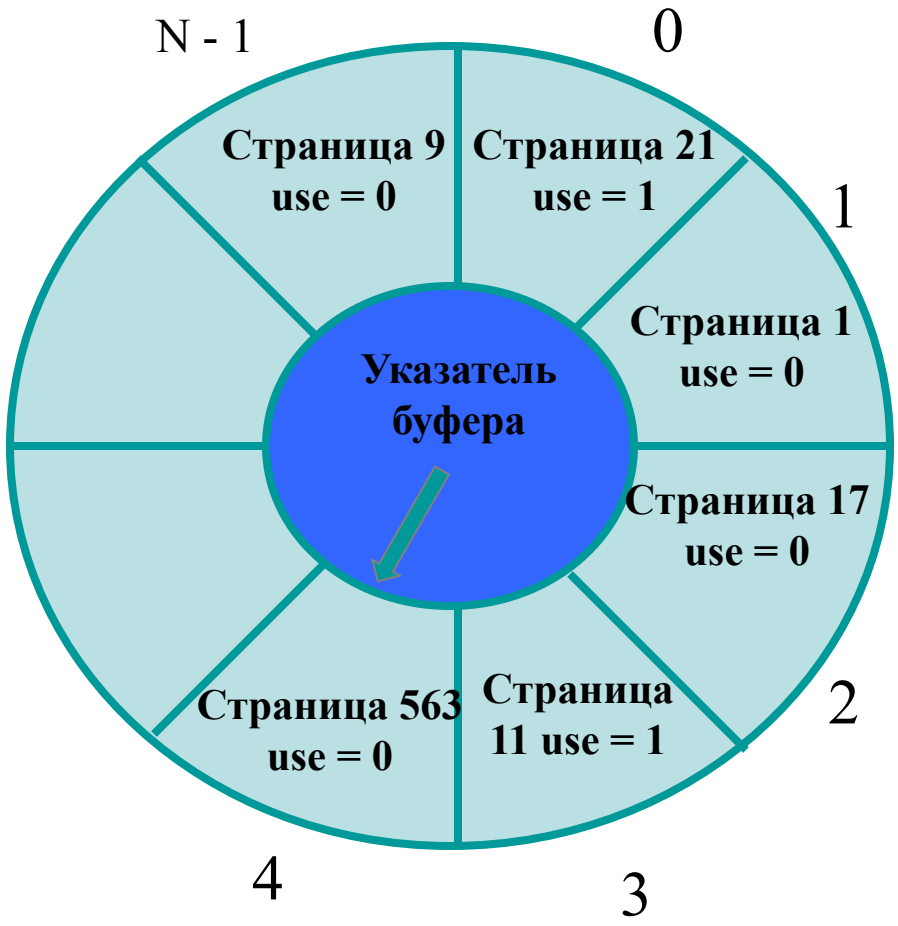
**Идеальная стратегия замещения: должна быть замещена та страница, к которой дольше всего не будет обращений в будущем.**

| НАИМЕНОВАНИЕ | ВОЗМОЖНЫЕ АЛГОРИТМЫ |
|---|---|
| Стратегия выборки (когда?) | По требованию, предварительная выборка |
| Стратегия размещения (где?) | Первый подходящий раздел для сегментной виртуальной памяти. Любая страница физической памяти для сегментно-страничной и страничной организации памяти. |
| Стратегия замещения (какие?) | Оптимальный выбор, дольше всех не использовавшиеся, первым вошел – первым вышел (FIFO), часовой, буферизация страниц. |
| Управление резидентным множеством (сколько?) | Фиксированный размер, переменный размер, локальная и глобальная области видимости. |
| Стратегия очистки (когда?) | По требованию, предварительная очистка |
| Управление загрузкой (сколько?) и приостановкой процессов | Рабочее множество, критерии L = S (среднее время между прерываниями = среднему времени обработки прерывания) и 50% |

# Часовая стратегия замещения



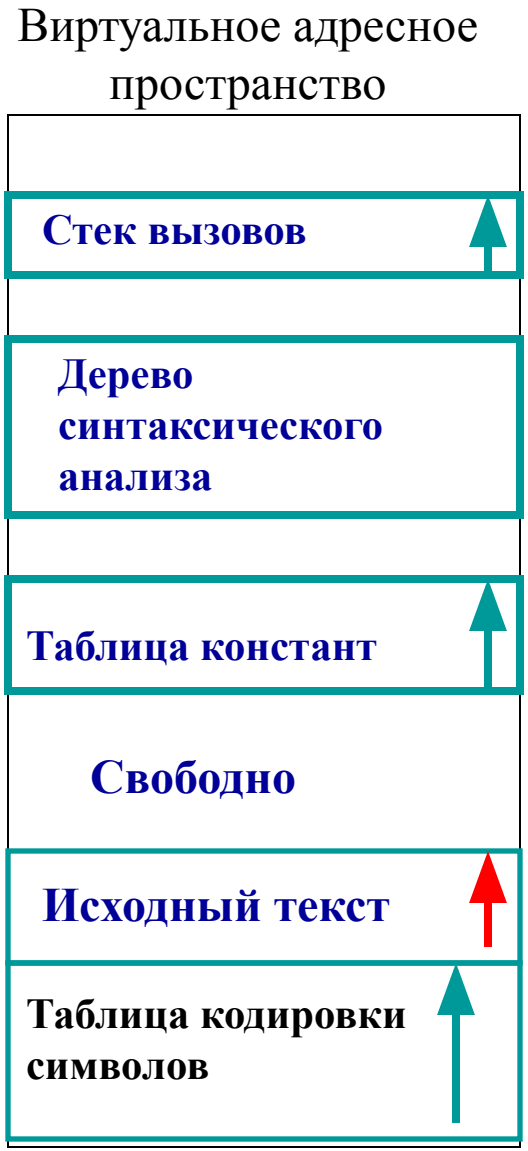

**Состояние буфера перед замещением страниц**




**Состояние буфера после замещения страниц**

- Имеется циклический буфер размерности n, в каждом элементе хранится номер страницы и бит использования. Бит использования установлен в 1, когда к странице произведено обращение. Указатель буфера указывает на последнюю замещенную страницу. Когда возникает необходимость решить задачу замещения, указатель перемещается на следующий элемент буфера. Если бит использования установлен в 0, то производится замещение соответствующей страницы. Если же окажется, что бит использования =1, то страница не замещается, бит использования устанавливается в 0, а указатель перемещается на следующий элемент буфера. Перемещение указателя осуществляется до тех пор, пока не будет обнаружена страница с 0 битом использования.

# Сегментная организация виртуальной памяти

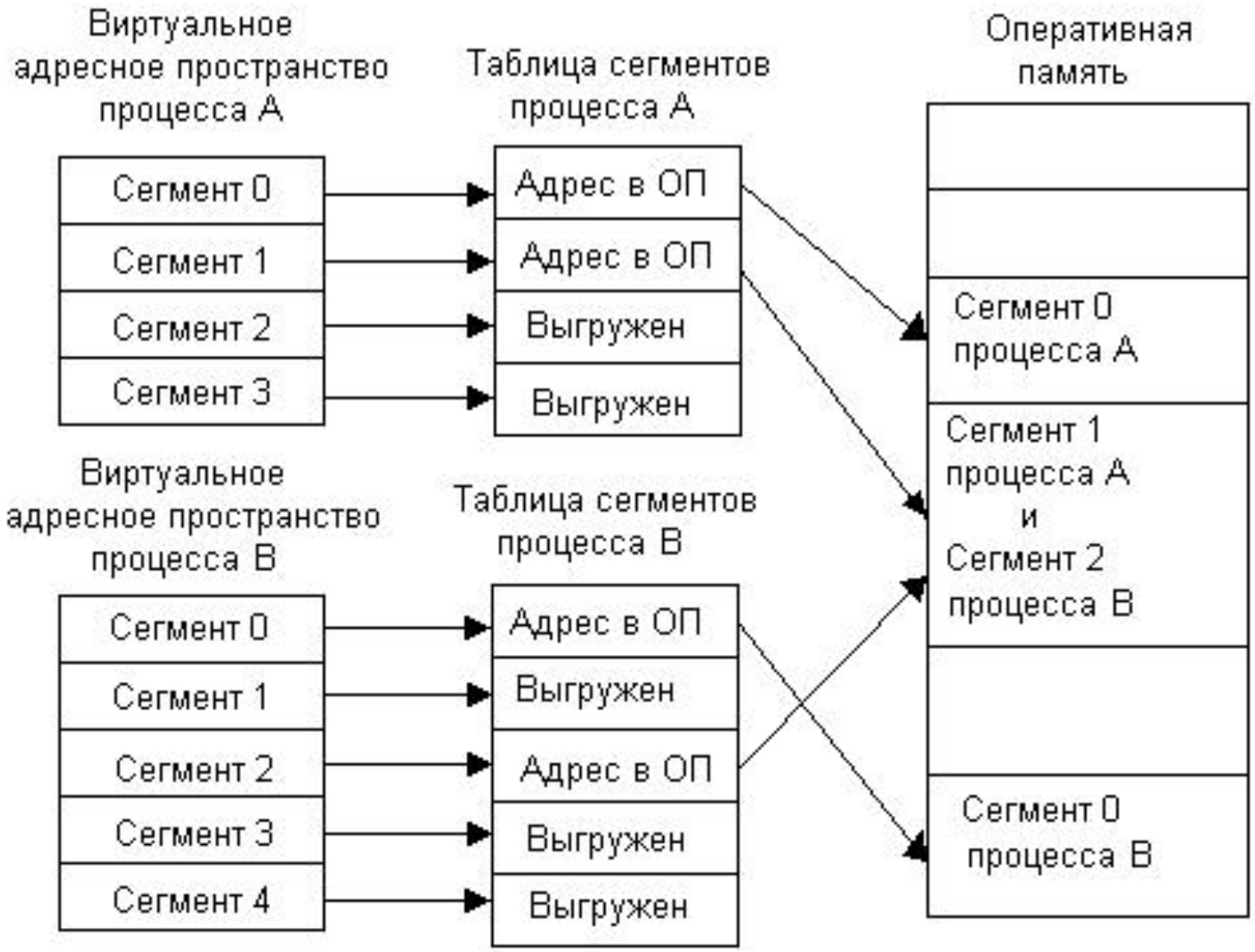



При компиляции возможно создание следующих сегментов:

1. Исходный текст, сохраненный для печати листинга программы.
2. Символьная таблица, содержащая имена и атрибуты переменных.
3. Таблица констант.
4. Дерево грамматического разбора, содержащее синтаксический анализ программы.
5. Стек, используемый для процедурных вызовов внутри компилятора.

**Таблица кодировки символов достигла таблицы с исходным текстом**

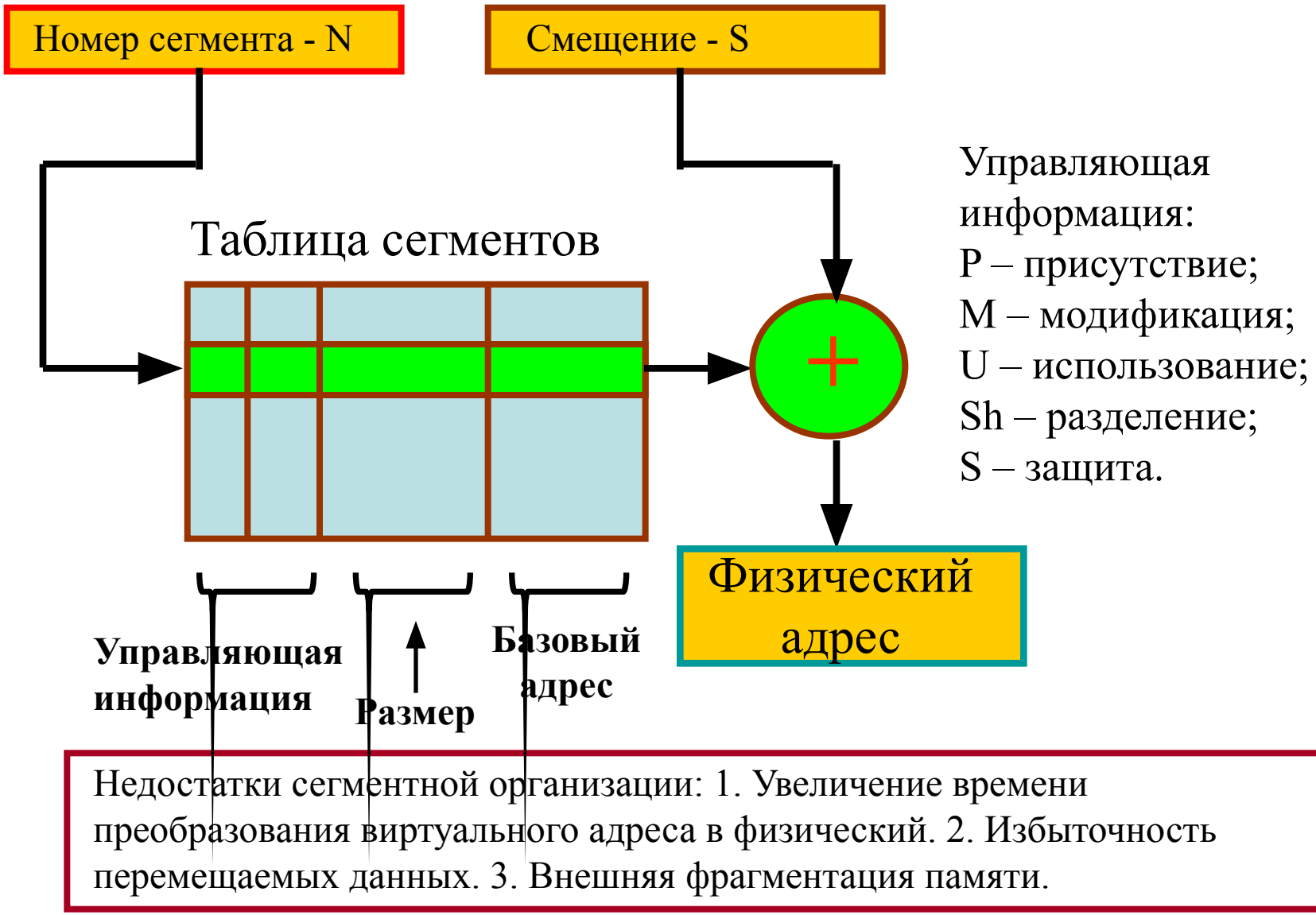
# Сегментная организация виртуальной памяти



# Сравнение страничной и сегментной организации памяти

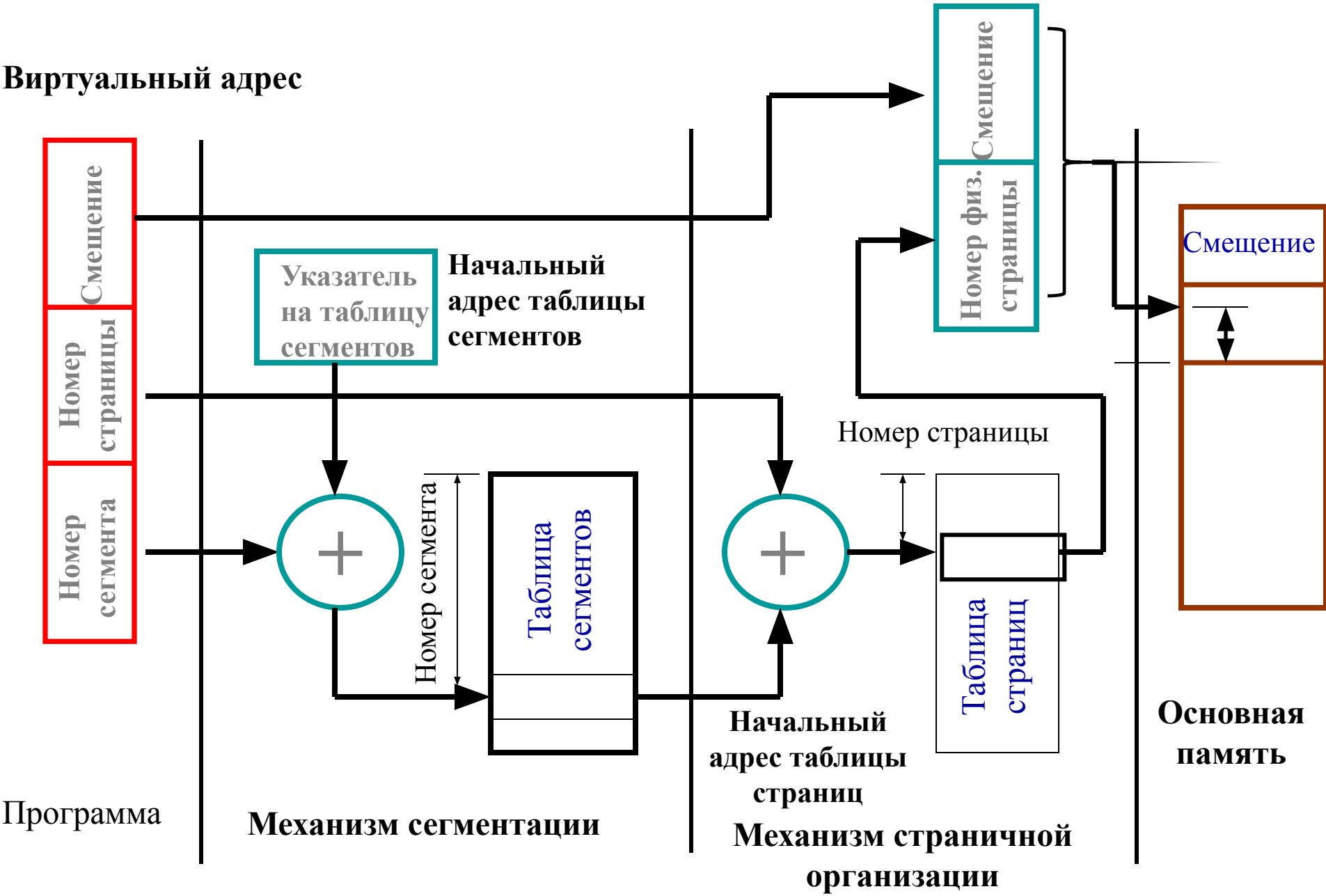
| Вопрос | Страничная | Сегментация |
|---|---|---|
| Нужно ли программисту знать о том, что используется эта техника? | Нет | Да |
| Сколько в системе линейных адресных пространств? | 1 | Много |
| Может ли суммарное адресное пространство превышать размеры физической памяти? | Да | Да |
| Возможно ли разделение процедур и данных, а также раздельная защита для них? | Нет | Да |
| Легко ли размещаются таблицы с непостоянными размерами? | Нет | Да |
| Облегчен ли совместный доступ пользователей к процедурам? | Нет | Да |
| Зачем была придумана эта техника? | Чтобы получить большое линейное адресное пространство без затрат на физическую память | Для разбиения программ и данных на независимые адресные пространства, облегчения защиты и совместного доступа |

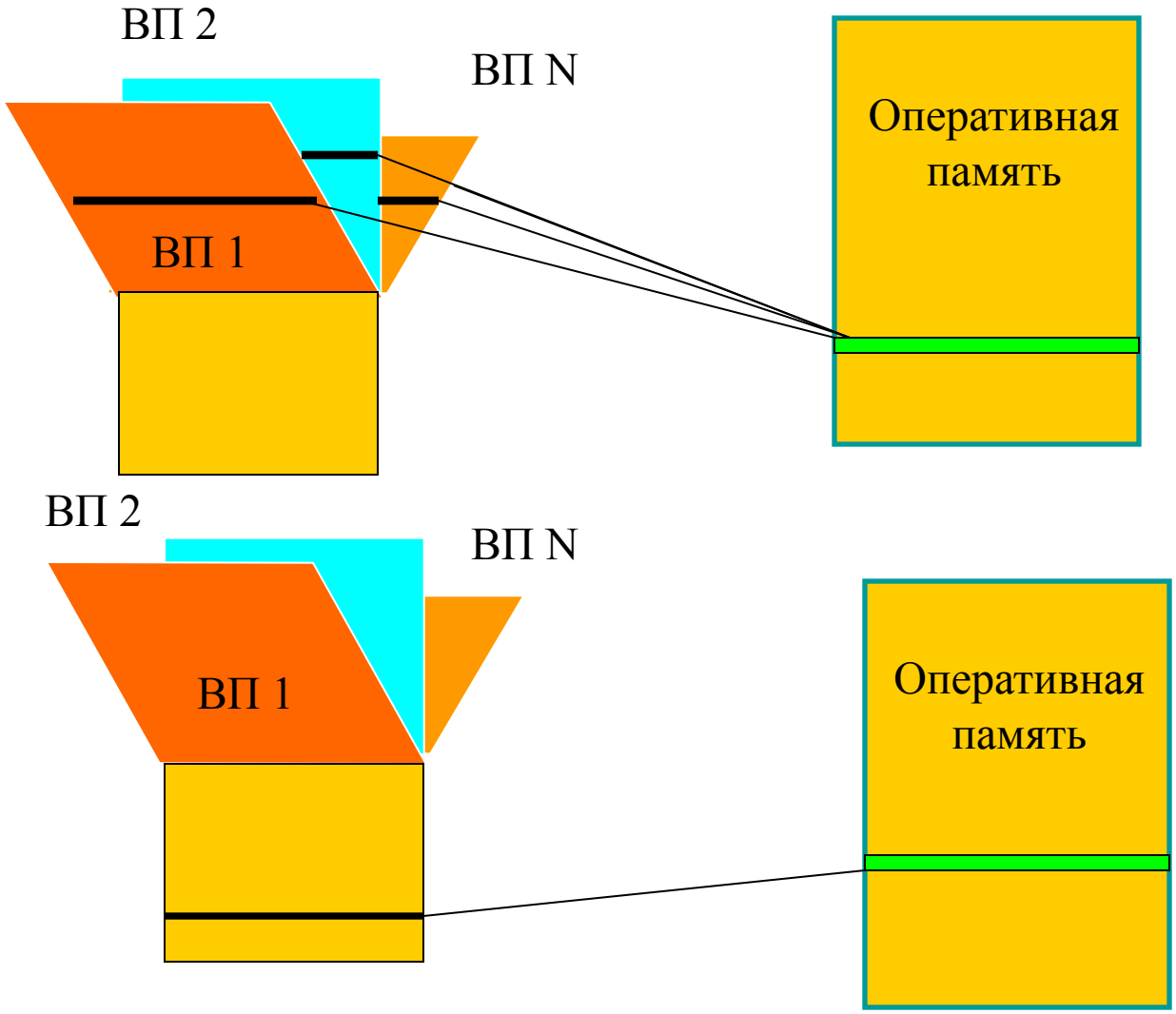
# Виртуальный адрес




Недостатки сегментной организации: 1. Увеличение времени преобразования виртуального адреса в физический. 2. Избыточность перемещаемых данных. 3. Внешняя фрагментация памяти.

# Сегментно-страничная организация виртуальной памяти



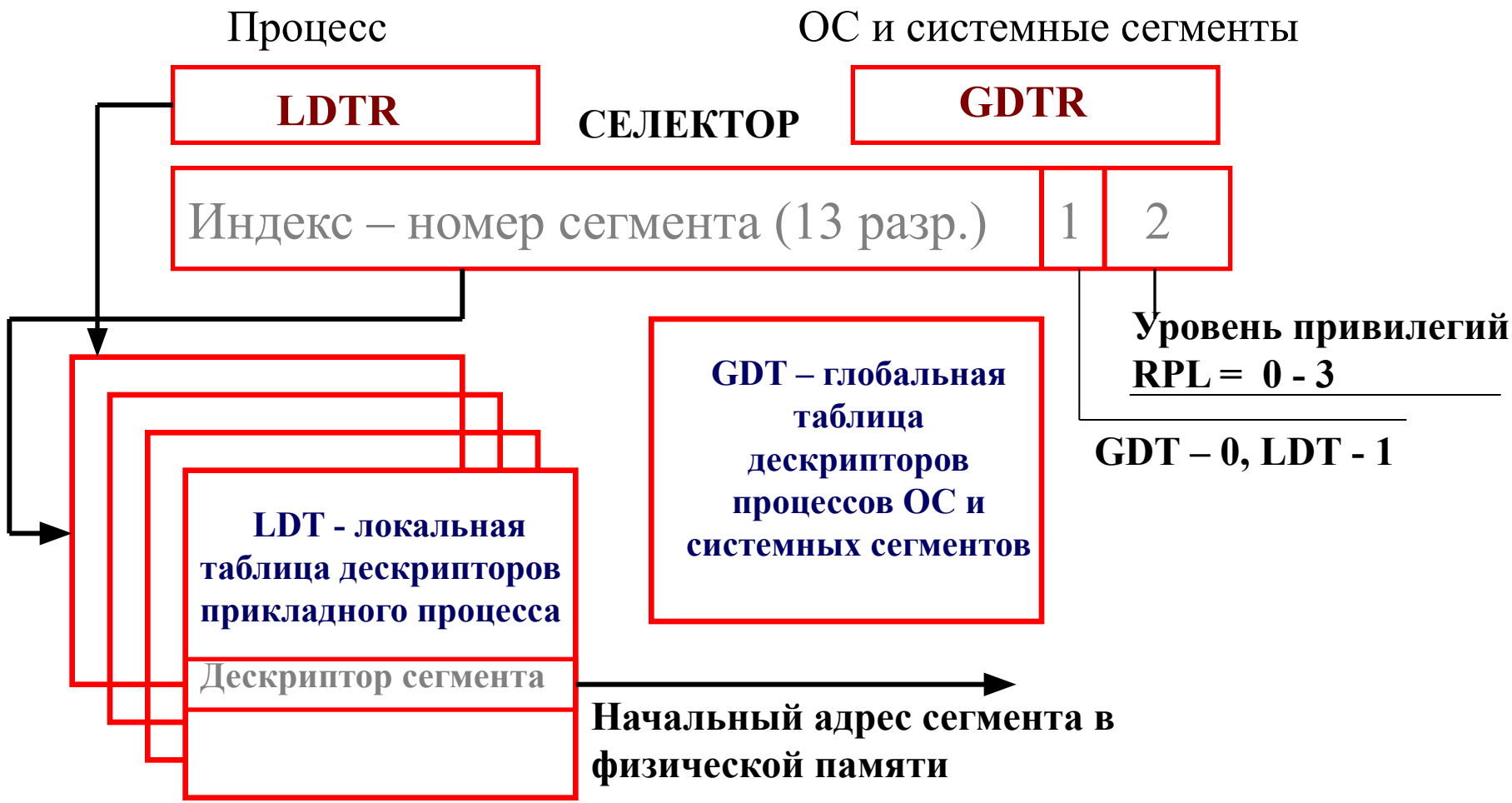


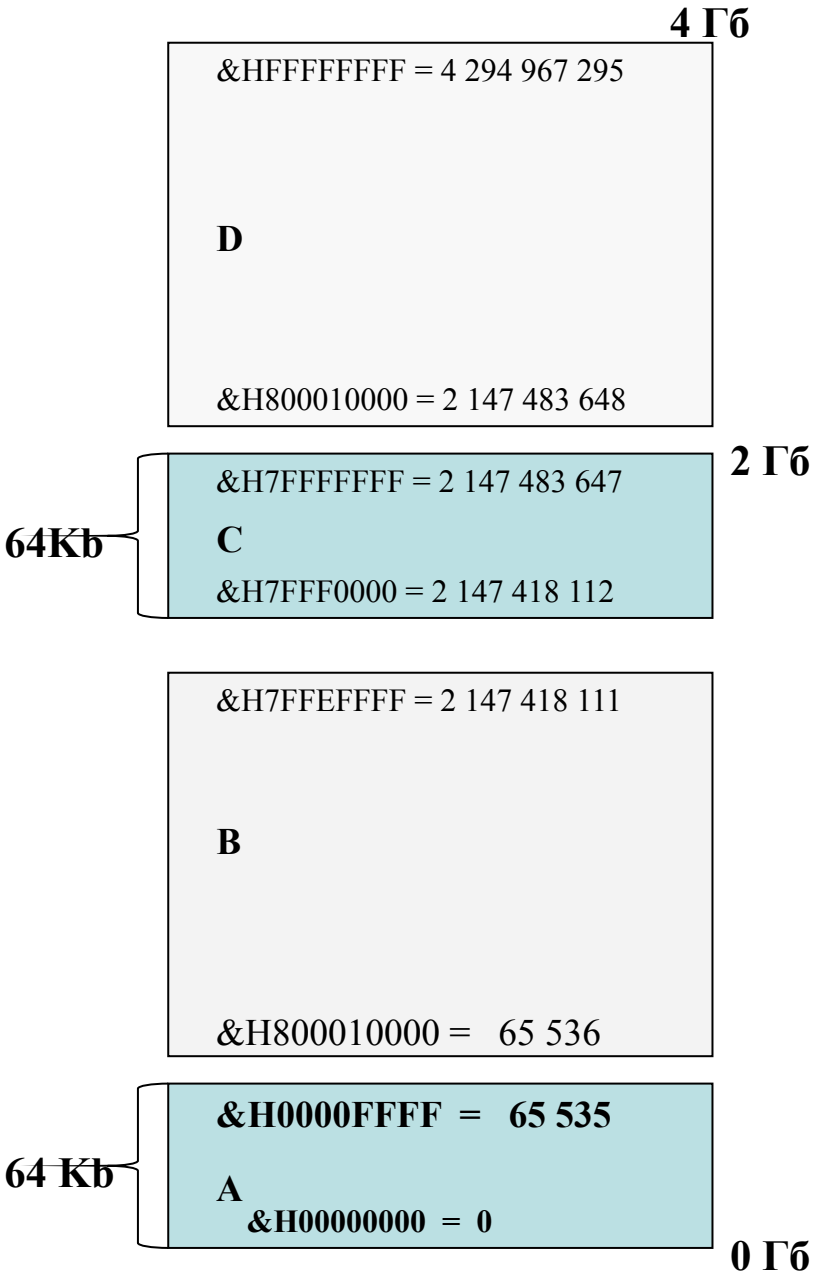

Способы создания разделяемого сегмента памяти

Виртуальная память Windows обеспечивает каждому процессу:

1. 4 Гбайт виртуального адресного пространства (2 Гбайт – ОС, 2 Гбайт – пользовательская программа).

2. 16 К независимых сегментов (8к локальных и 8К глобальных).



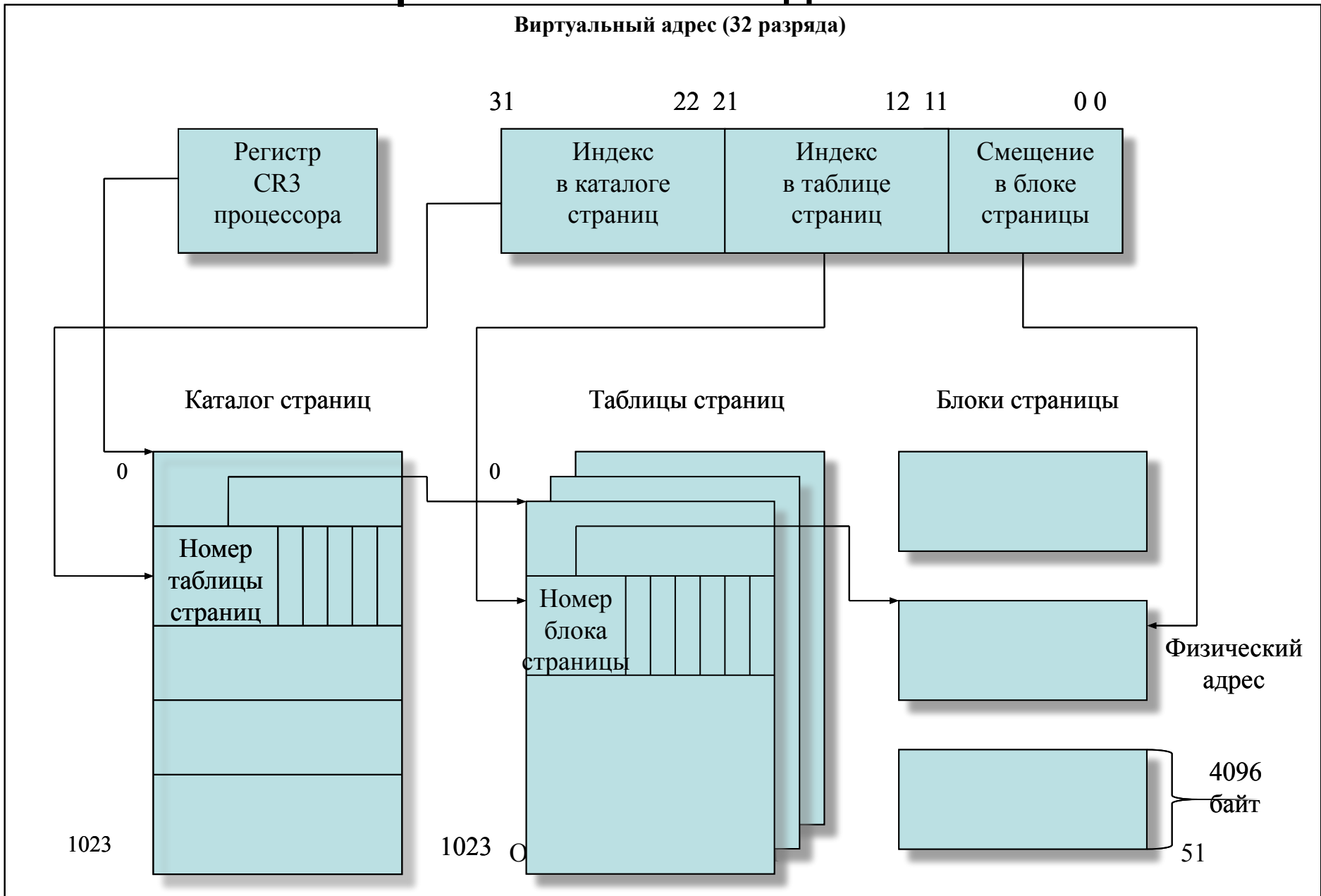


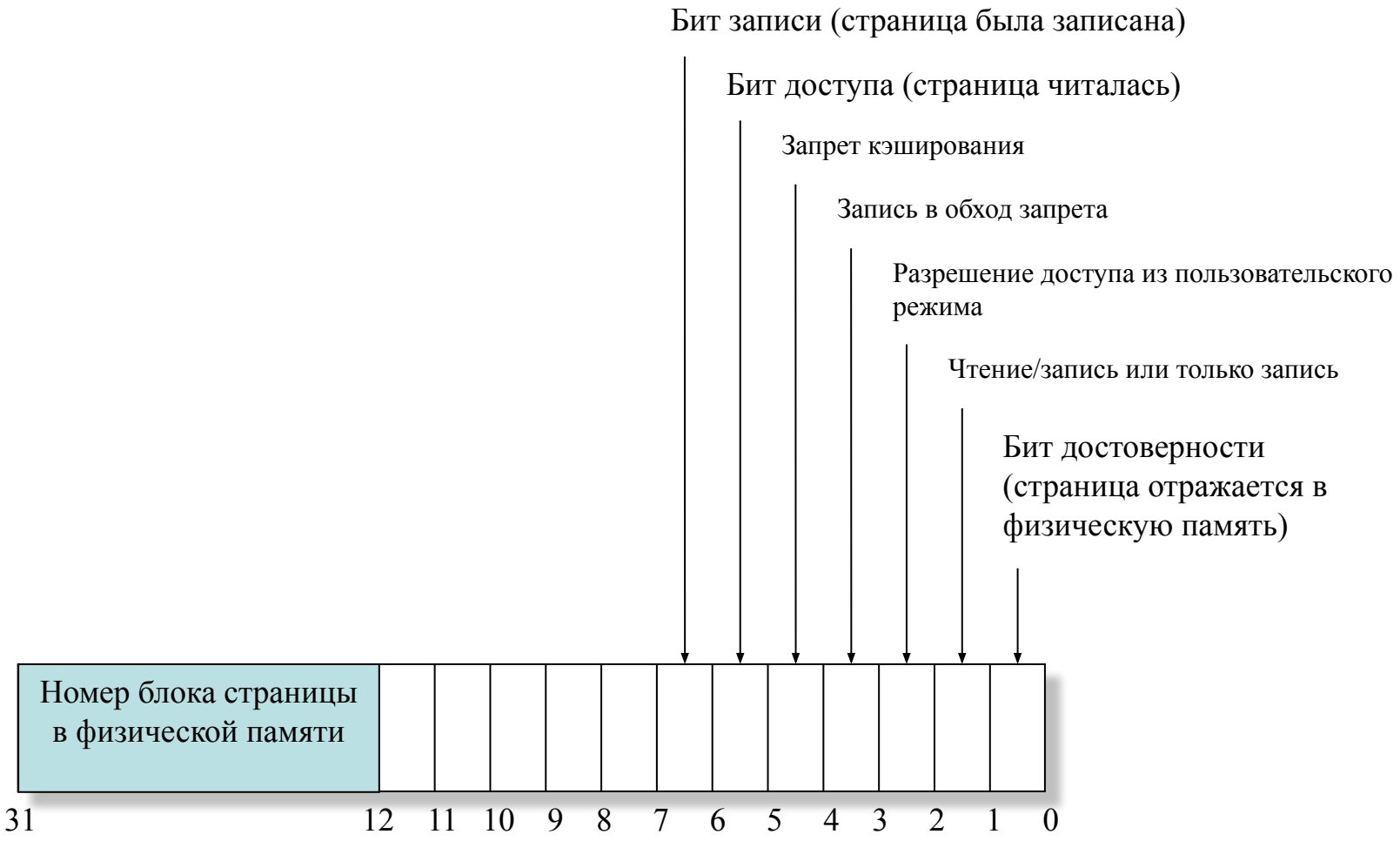

Зарезервировано Windows для исполнительной системы Windows, ядра и драйверов устройств.
Недоступно в пользовательском режиме.
(2 Гб)

Используется для некорректно инициализированных указателей.
Недоступно в пользовательском режиме. (64 Кб)

Адресное пространство процессов содержит прикладные модули EXE и DLL, Win32 DLL (kernel32.dll, user.dll и т.д.), файлы, отображаемые в память.
Доступно в пользовательском режиме.
(2 Гб – 128 Кб)

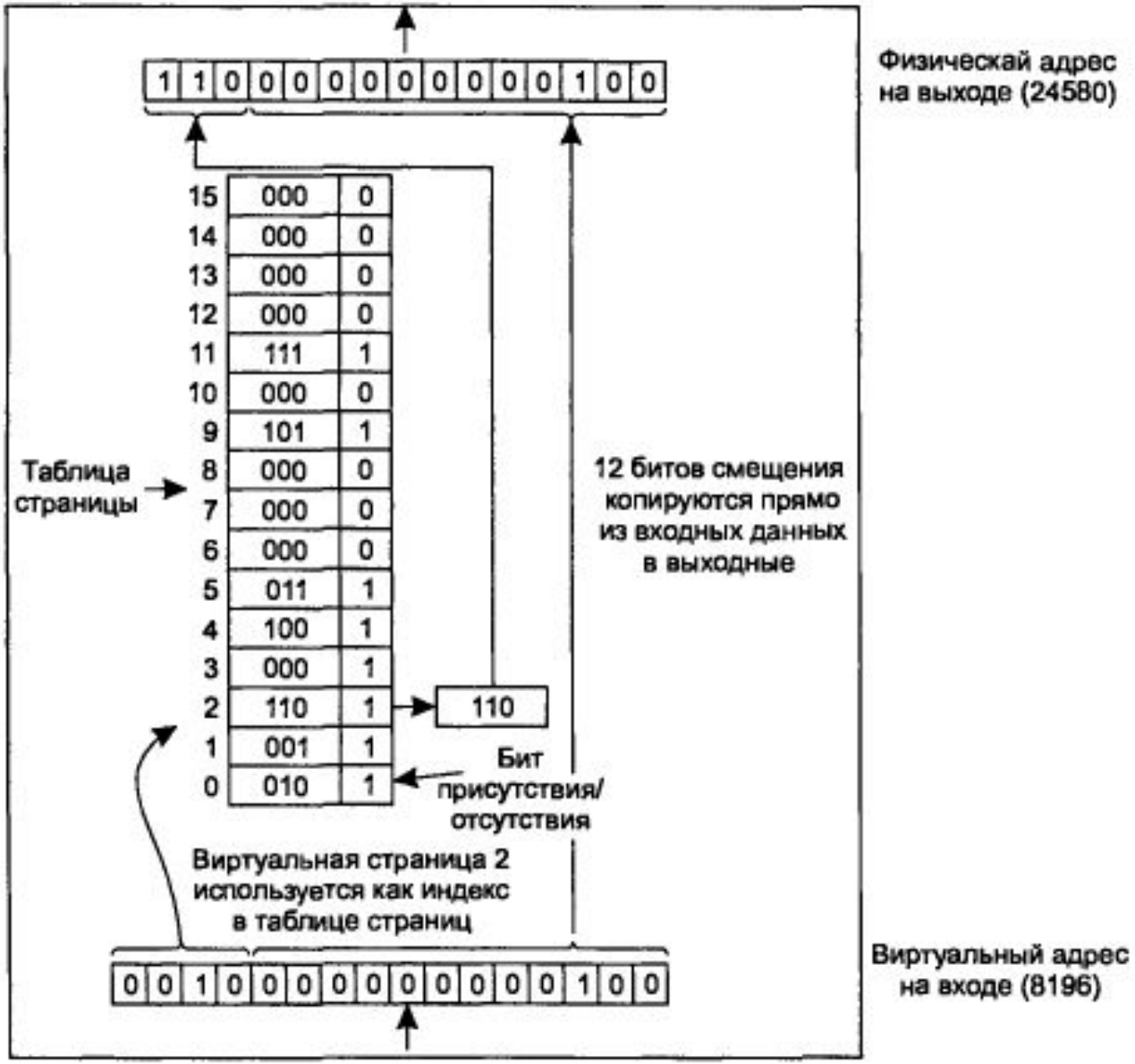
# Преобразование виртуальных адресов в физические: попадание




Бит записи (страница была записана)
Бит доступа (страница читалась)
Запрет кэширования
Запись в обход запрета
Разрешение доступа из пользовательского режима
Чтение/запись или только запись
Бит достоверности (страница отражается в физическую память)
Номер блока страницы в физической памяти
31
12
11
10
9
8
7
6
5
4
3
2
1
0

# *Защита памяти*

- PAGE_READONLY присваивает доступ «только для чтения» выделенной виртуальной памяти;
- PAGE_READWRITE назначает доступ «чтение-запись» выделенной виртуальной памяти;
- PAGE_WRITECOPY устанавливает доступ «запись копированием» (copy-onwrite) выделенной виртуальной памяти.
- PAGE_EXECUTE разрешает доступ «выполнение» выделенной виртуальной памяти. Тем не менее, любая попытка чтения - записи этой памяти приведет к нарушению доступа;
- PAGE_EXECUTE_READ назначает доступ «выполнение» и «чтение»;
- PAGE_EXECUTE_READWRITE разрешает доступ «выполнение», «чтение» и «запись»;
- PAGE_EXECUTE_WRITECOPY присваивает доступ «выполнение», «чтение» и «запись копированием»;
- PAGE_NOACCESS запрещает все виды доступа к выделенной виртуальной памяти.




Внутренняя операция диспетчера памяти в системе
с шестнадцатью страницами размером 4 Кбайт

# Размещение объектов в памяти

- Регионом, областью или блоком памяти называют непрерывную последовательность ячеек памяти. Обычно регион больше области, а область больше блока.
- В общем случае оперативная память, с которой работает программа, подразделяется на три вида: ***статическую***, ***автоматическую*** и ***динамическую***.
- Статическая память — это область памяти, выделяемая при запуске программы до вызова функции main из свободной оперативной памяти для размещения глобальных и статических объектов, а также объектов, определённых в пространствах имён.

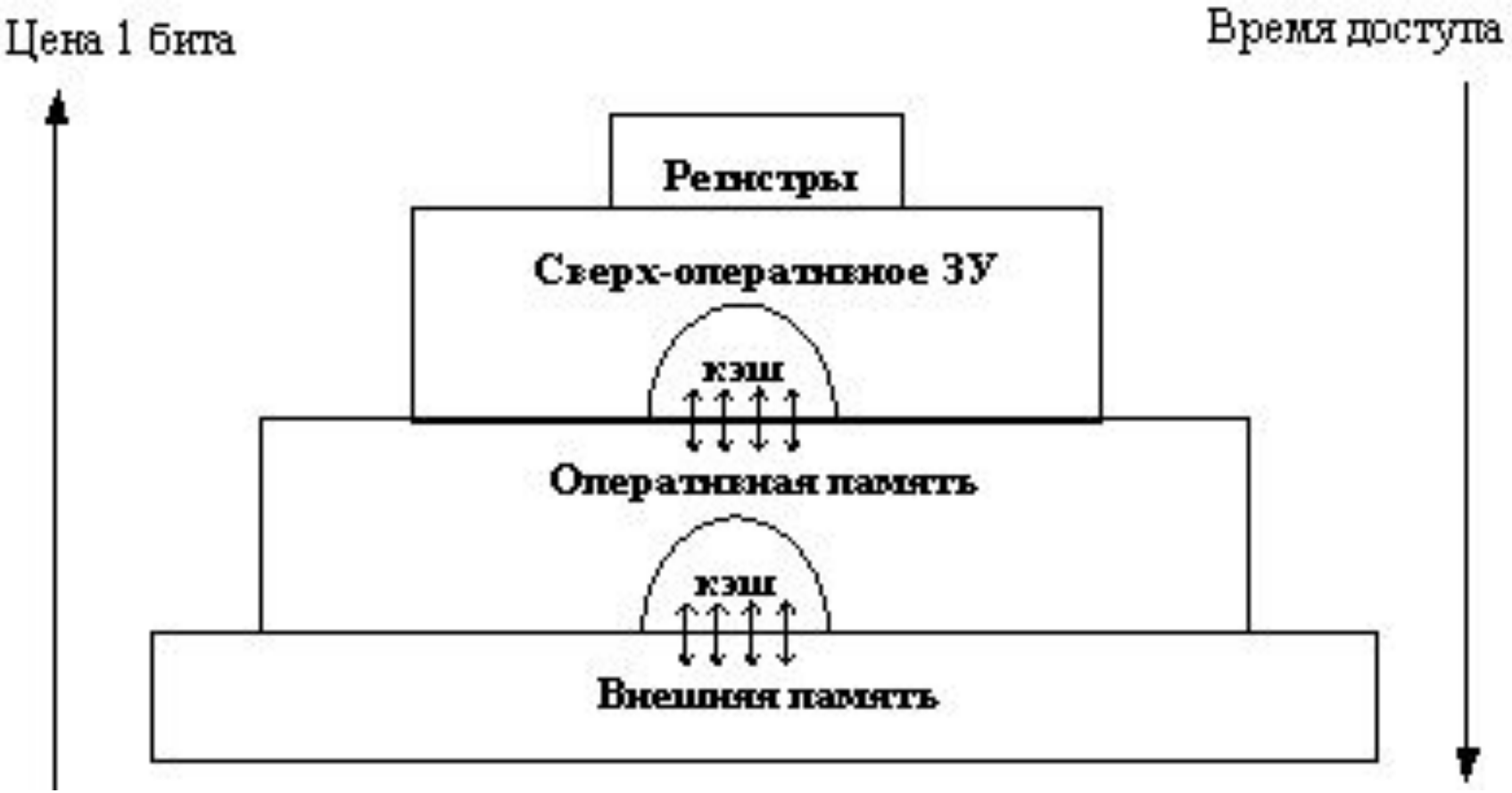
**Автоматическая память** — это специальный регион памяти, резервируемый при запуске программы до вызова функции main из свободной оперативной памяти и используемый в дальнейшем для размещения локальных объектов: объектов, определяемых в теле функций и получаемых функциями через параметры в момент вызова. Автоматическую память часто называют **стеком**.

**Динамическая память** — это совокупность блоков памяти, выделяемых из доступной свободной оперативной памяти непосредственно во время выполнения программы под размещение конкретных объектов.

- Доступную программе свободную оперативную память часто, в связи со спецификой размещения в ней объектов, называют кучей (heap). В общем случае нет чёткого, наперёд заданного (как в стеке) порядка расположения объектов в ней: распределение блоков памяти происходит динамически — в большинстве реализаций под объект отводится первый подходящий по размеру свободный блок.
- В отличие от стека, которым управляет компилятор, управление динамической памятью осуществляется явным образом: выделение памяти производится оператором new, освобождение — оператором delete.

# *Кэш-память*

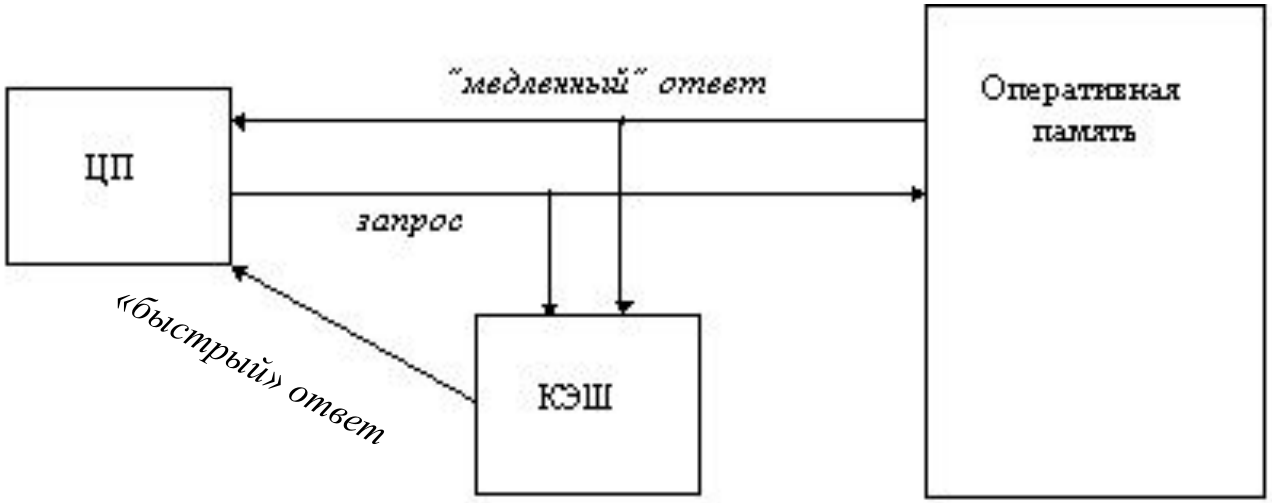
- *Кэш-память,* или просто *кэш (cache),* — это способ совместного функционирования двух типов запоминающих устройств, отличающихся временем доступа и стоимостью хранения данных, который за счет динамического копирования в «быстрое» ЗУ наиболее часто используемой информации из «медленного» ЗУ позволяет уменьшить среднее время доступа к данным.


Цена 1 бита
Время доступа
Регистры
Сверх-оперативное ЗУ
кэш
Оперативная память
кэш
Внешняя память

Содержимое кэш-памяти представляет собой совокупность *записей* обо всех загруженных в нее элементах данных из основной памяти.

Каждая запись об элементе данных включает в себя:

- значение элемента данных;
- адрес, который этот элемент данных имеет в основной памяти;
- дополнительную информацию, которая используется для реализации алгоритма замещения данных в кэше и обычно включает признак модификации и признак действительности данных.




Структура кэш-памяти

| Адрес данных в ОП | Данные | Управл. информация | |
|---|---|---|---|
| | | бит модиф. | бит обращ. |
| | | | |
| | | | |
| | | | |
| | | | |

Кэш-память не является адресуемой, поэтому поиск нужных данных осуществляется по содержимому — по взятому из запроса значению поля адреса в оперативной памяти.

Далее возможен один из двух вариантов развития событий:

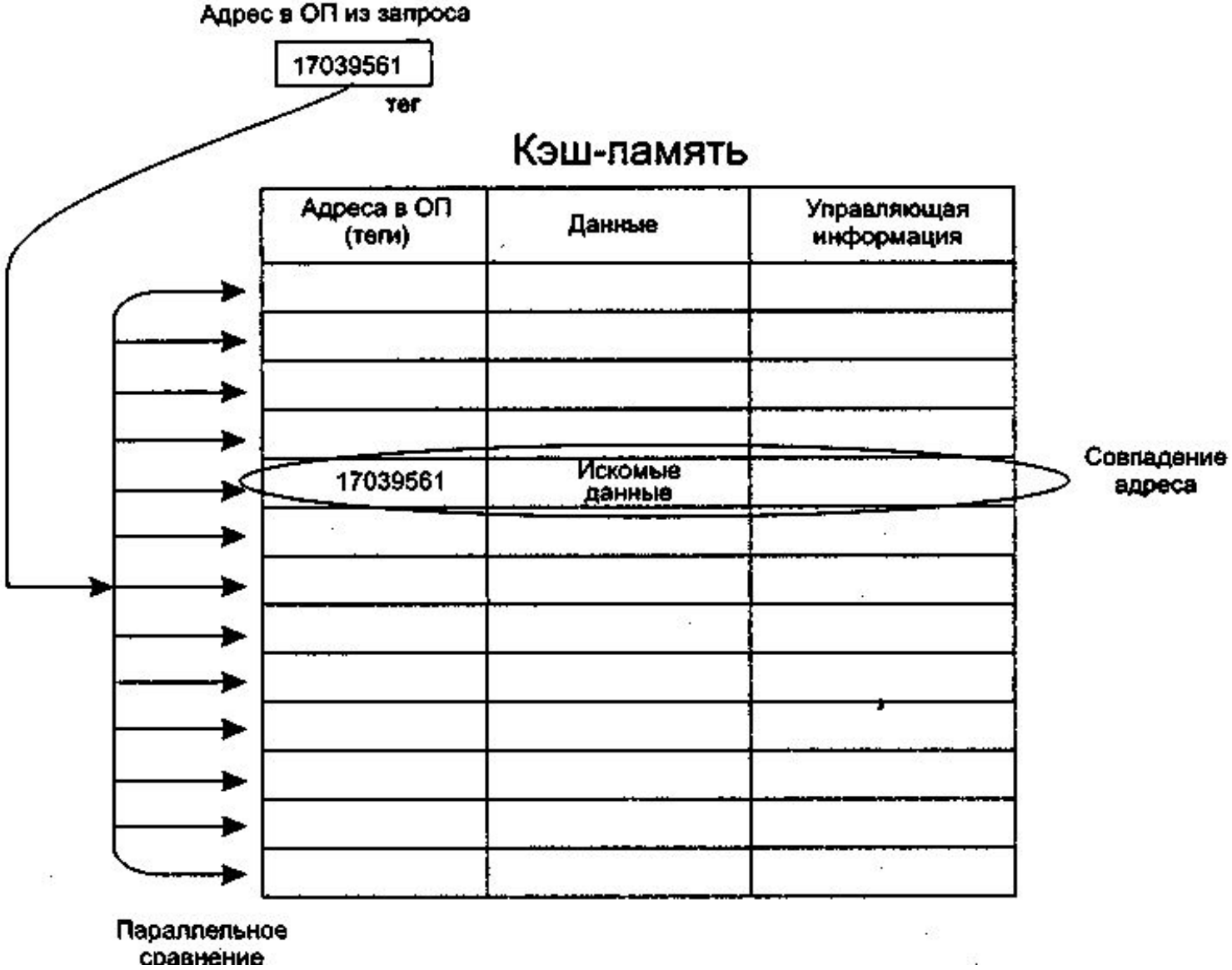
1. если данные обнаруживаются в кэш-памяти, то есть произошло *кэш-попадание (cache-hit),* они считываются из нее и результат передается источнику запроса;

2. если нужные данные отсутствуют в кэш-памяти, то есть произошел *кэш-промах (cache-miss),* они считываются из основной памяти, передаются источнику запроса и одновременно с этим копируются в кэш-память.

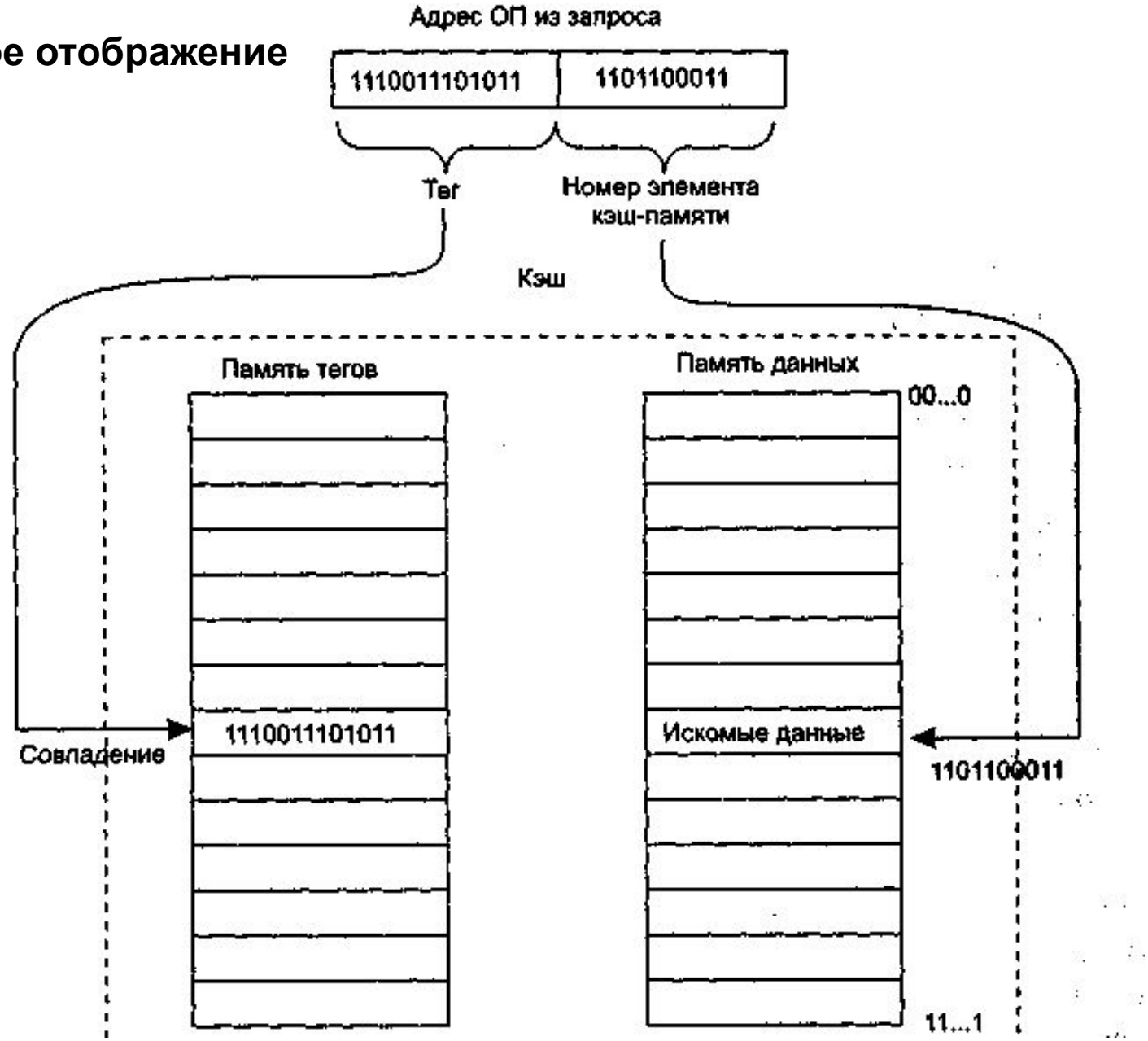
Наличие в компьютере двух копий данных — в основной памяти и в кэше — порождает проблему согласования данных.

- *Рассмотрим два подхода к решению этой проблемы:*
- *Сквозная запись (write through).*
- *Обратная запись (write back).*

# Ассоциативный поиск в кэше со случайным отображением



## Прямое отображение



**Комбинирование прямого и случайного отображения**

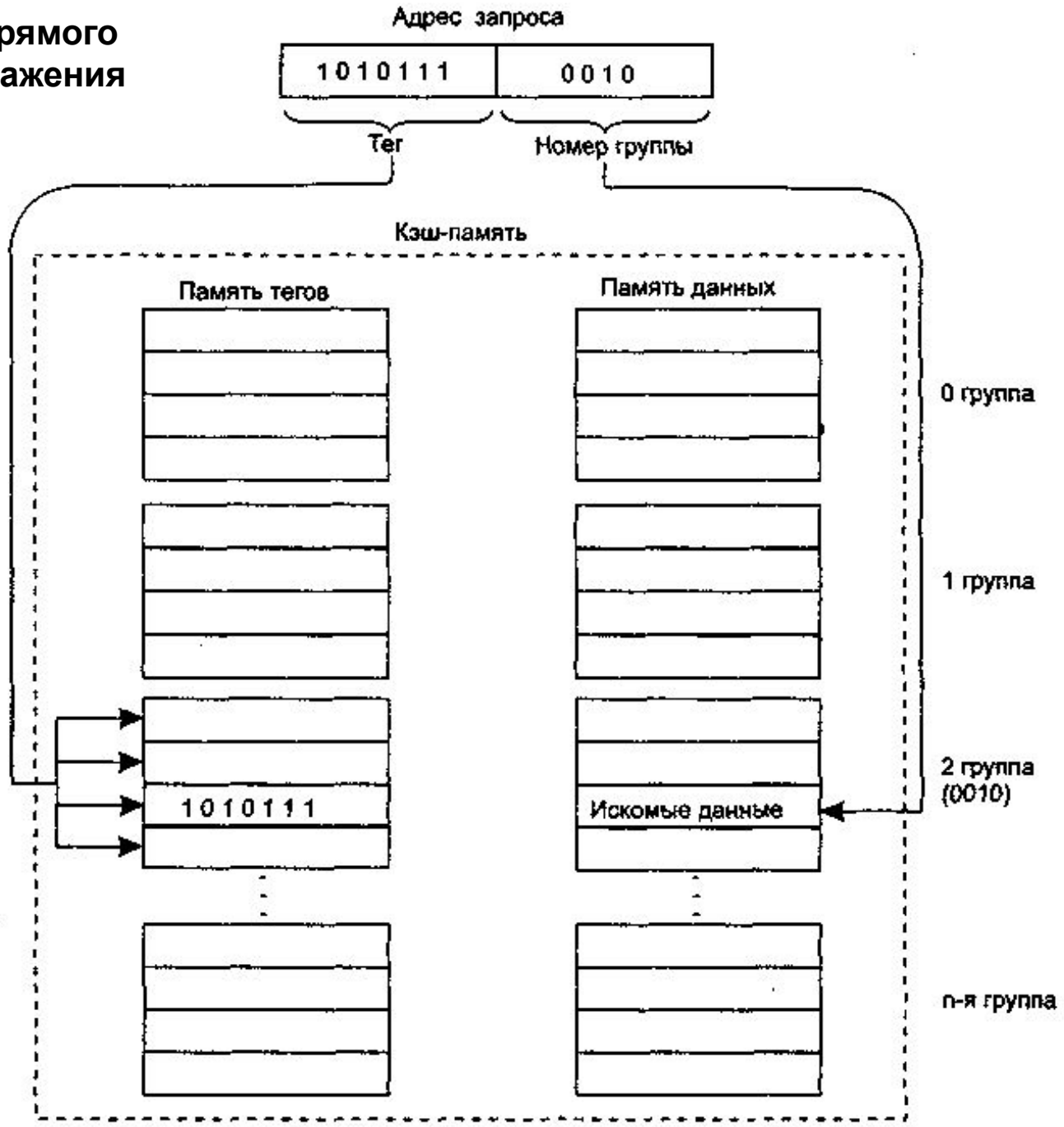


**Схема выполнения запроса на чтение в системе с двухуровневым кэшем**

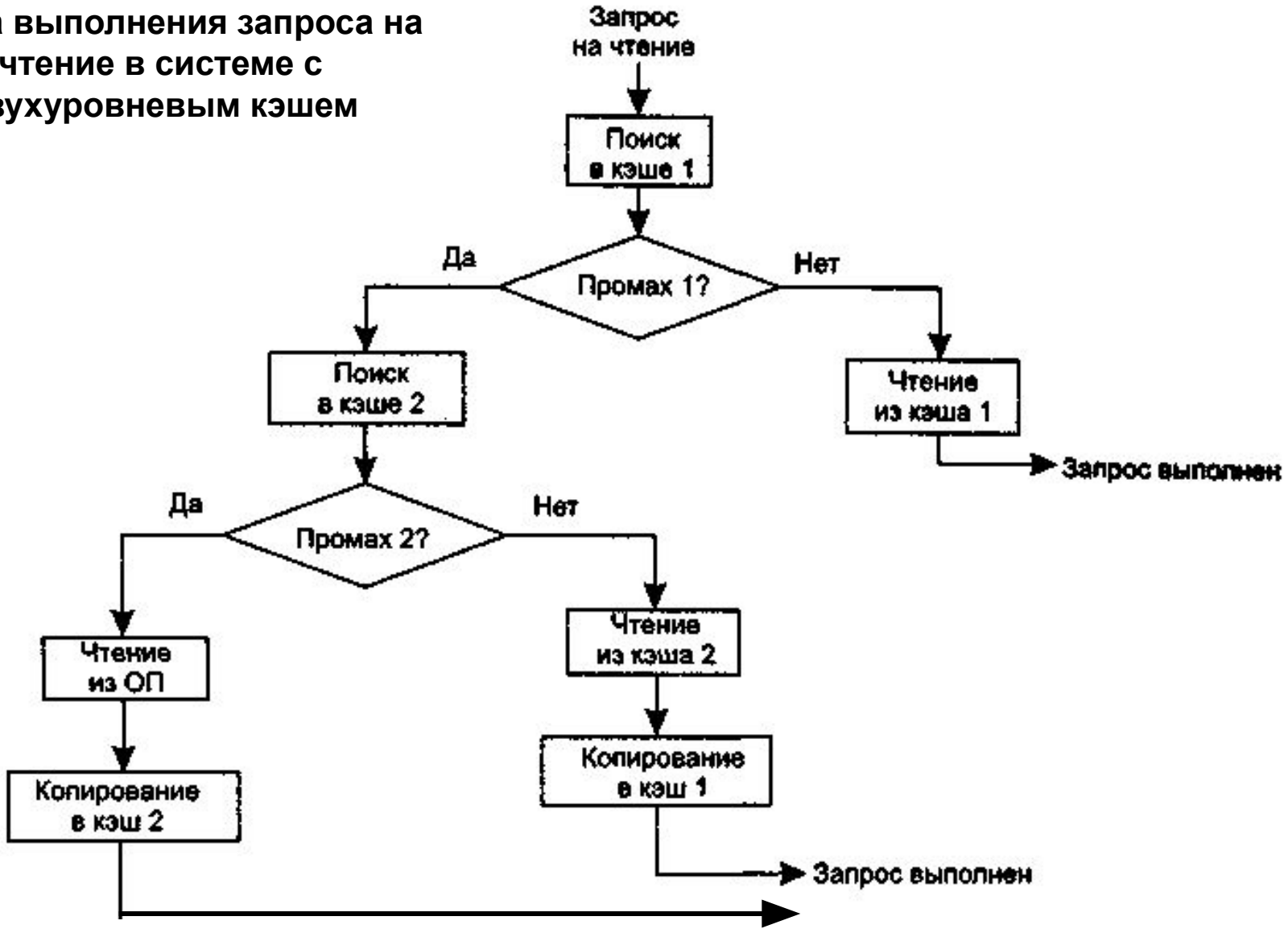
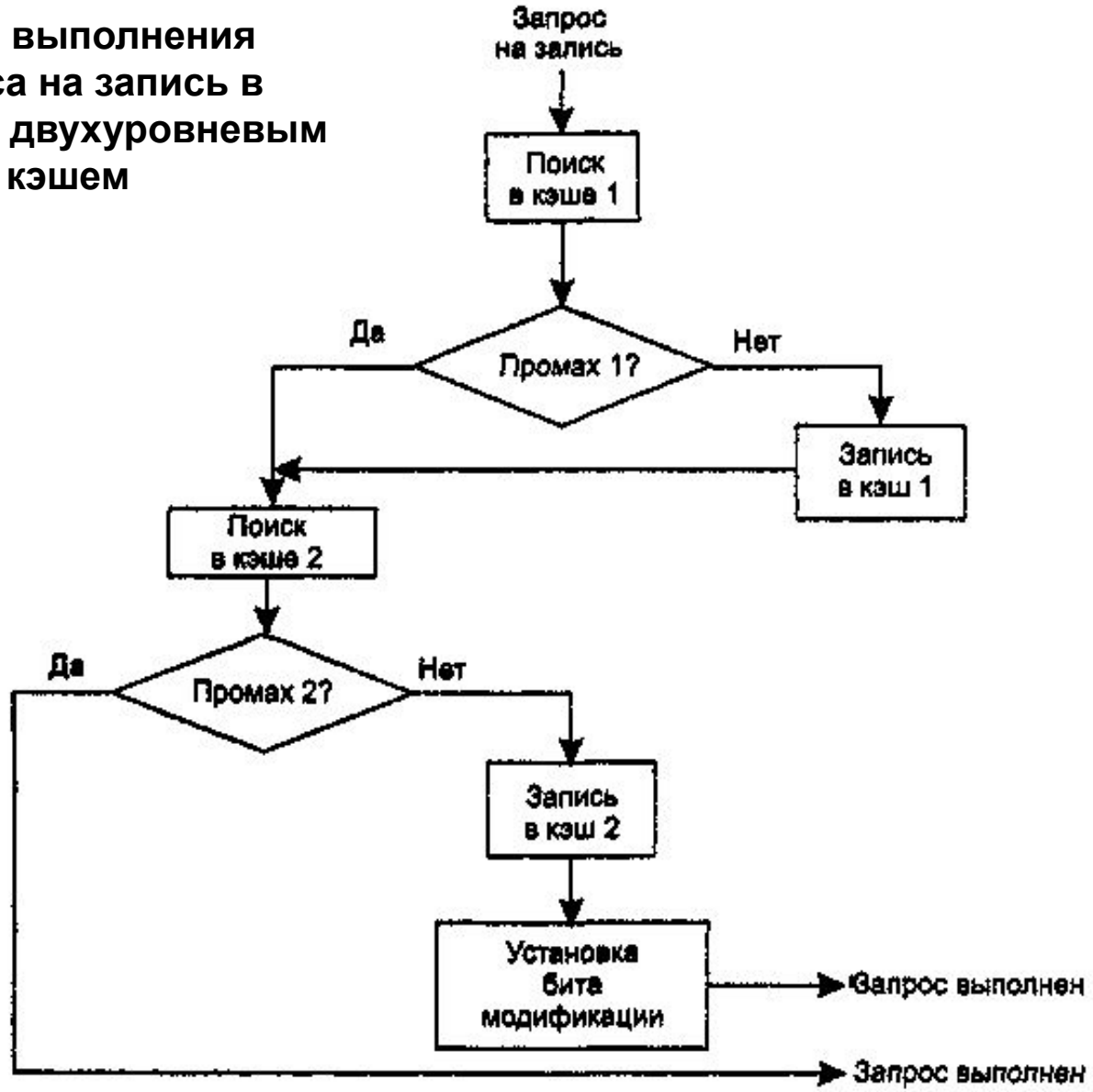
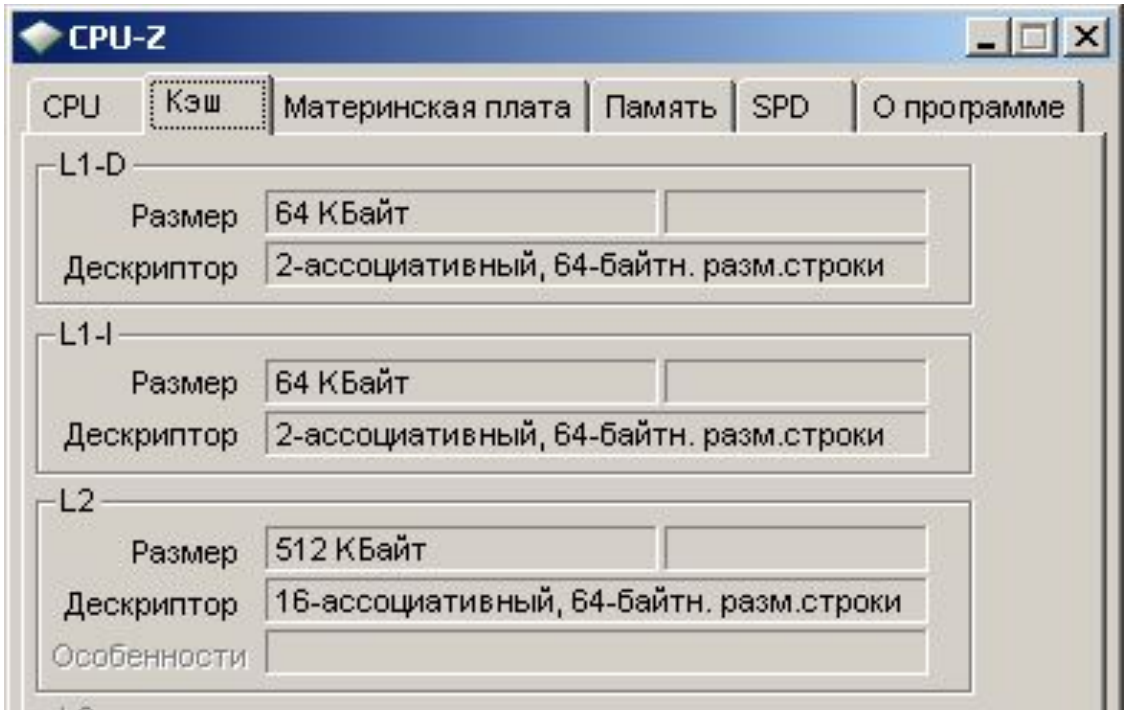


**Схема выполнения запроса на запись в системе с двухуровневым кэшем**



# Анализ процессора и памяти




Самой быстрой памятью является кэш первого уровня — L1-cache. По сути, она является неотъемлемой частью процессора, поскольку расположена на одном с ним кристалле и входит в состав функциональных блоков. Состоит из кэша команд и кэша данных. Латентность доступа обычно равна 2−4 тактам ядра.

Вторым по быстродействию является L2-cache — кэш второго уровня. Обычно он расположен либо на кристалле, как и L1, либо в непосредственной близости от ядра, например, в процессорном картридже. Обычно латентность L2 кэша, расположенного на кристалле ядра, составляет от 8 до 20 тактов ядра.

Кэш третьего уровня наименее быстродействующий и обычно расположен отдельно от ядра ЦП, но он может быть очень внушительного размера — более 32 Мбайт. L3 кэш медленнее предыдущих кэшей, но всё равно значительно быстрее, чем оперативная память.