Операционные системы

Тема 3. Управление памятью. Методы, алгоритмы и средства

Тема 3. Управление памятью. Методы, алгоритмы и средства

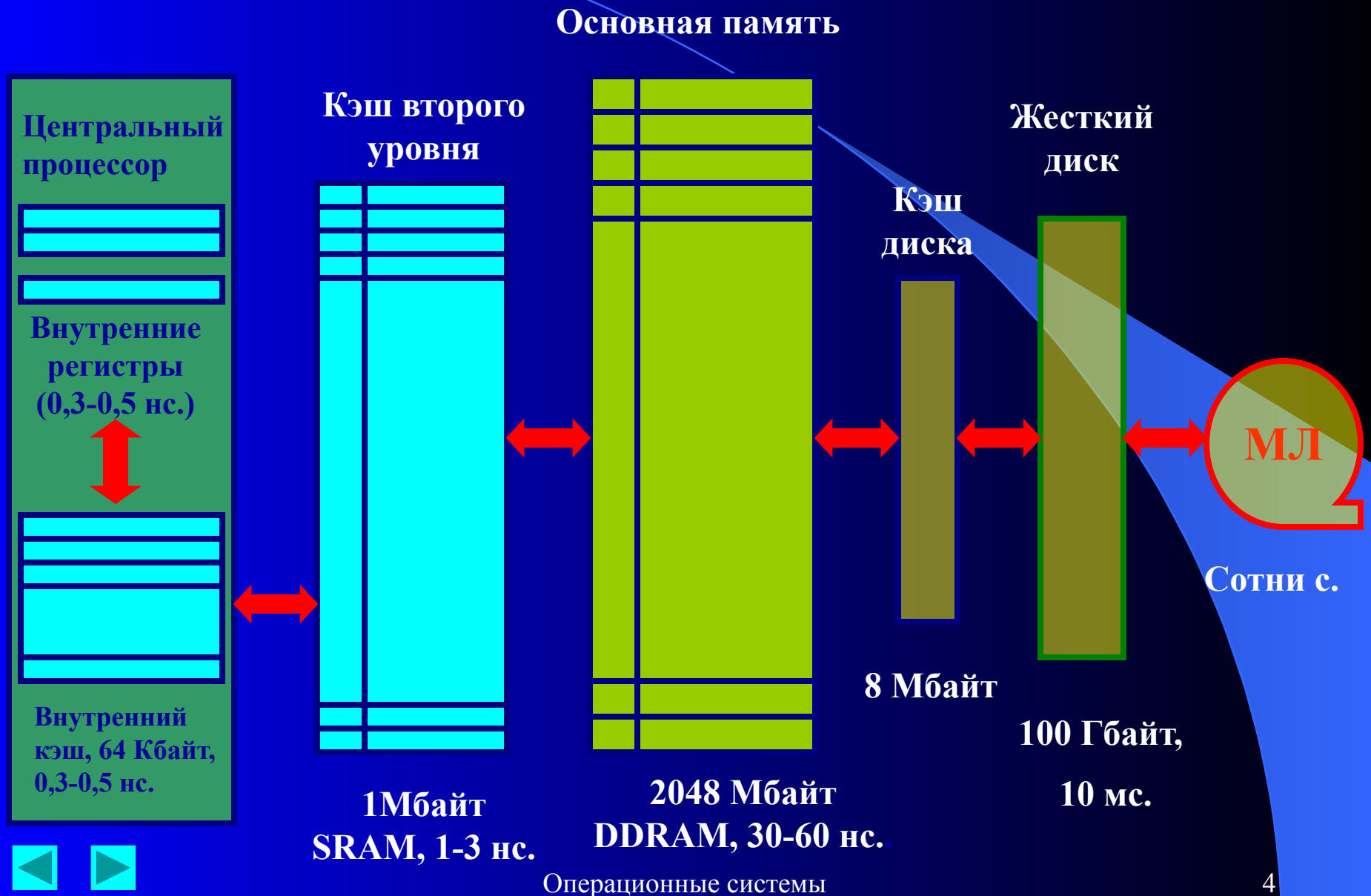
3.1. Организация памяти современного компьютера

3.1.1. Логическая организация памяти: Линейное (одномерное) адресное пространство, отражающее особенности аппаратного обеспечения, но не соответствующее современной технологии создания программного обеспечения.

Для эффективной работы с пользовательскими программами необходимо чтобы:

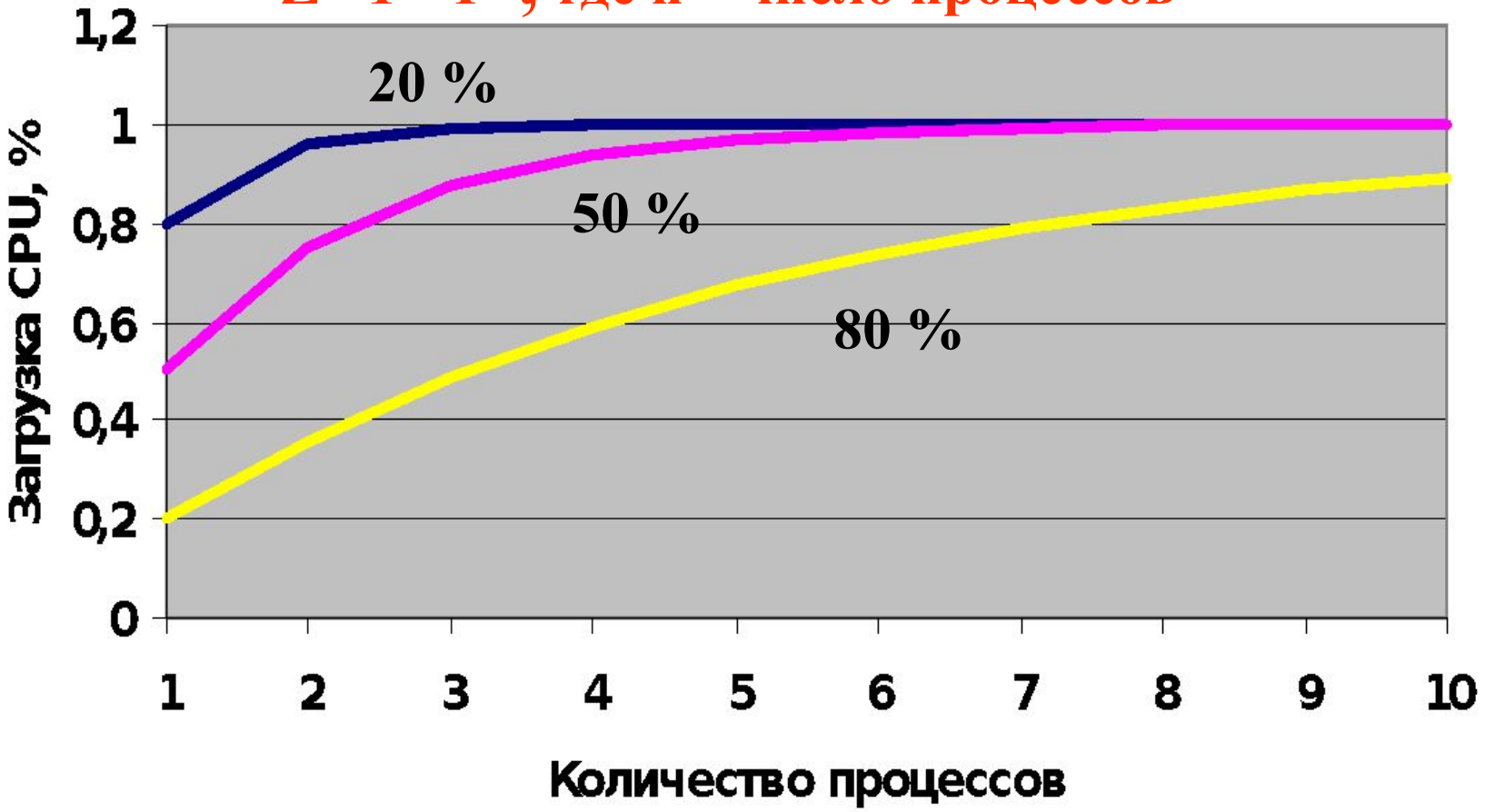
- Модули могли быть созданы и скомпилированы независимо друг от друга, при этом все ссылки из одного модуля в другой разрешаются системой во время работы программы.
- Разные модули могли получать разные степени защиты (только чтение, только исполнение и т. п.) за счет весьма умеренных накладных расходов.
- Возможно применение механизма, обеспечивающего совместное использование модулей разными процессами (для случая сотрудничества разных процессов в работе над одной задачей).

3.1.2. Физическая организация памяти



Зависимость загрузки процессора от количества процессов в памяти

$Z = 1 - P^n$, где n – число процессов



3.1.3. Виртуальная память

Виртуализация оперативной памяти осуществляется совокупностью аппаратных и программных (ОС) средств вычислительной системы автоматически без участия программиста и не сказывается на работе приложения.

Методы виртуализации памяти: свопинг (swapping), виртуальная память (virtual memory).

Достоинства свопинга: малые затраты времени на преобразование адресов в кодах программ. **Недостатки**: избыточность перемещаемых данных, замедление работы системы, неэффективное использование памяти, невозможность загрузить процесс, адресное пространство которого превышает объем свободной оперативной памяти.

Недостатки виртуальной памяти: необходимость преобразования виртуальных адресов в физические, сложность аппаратной и программной (ОС) поддержки.

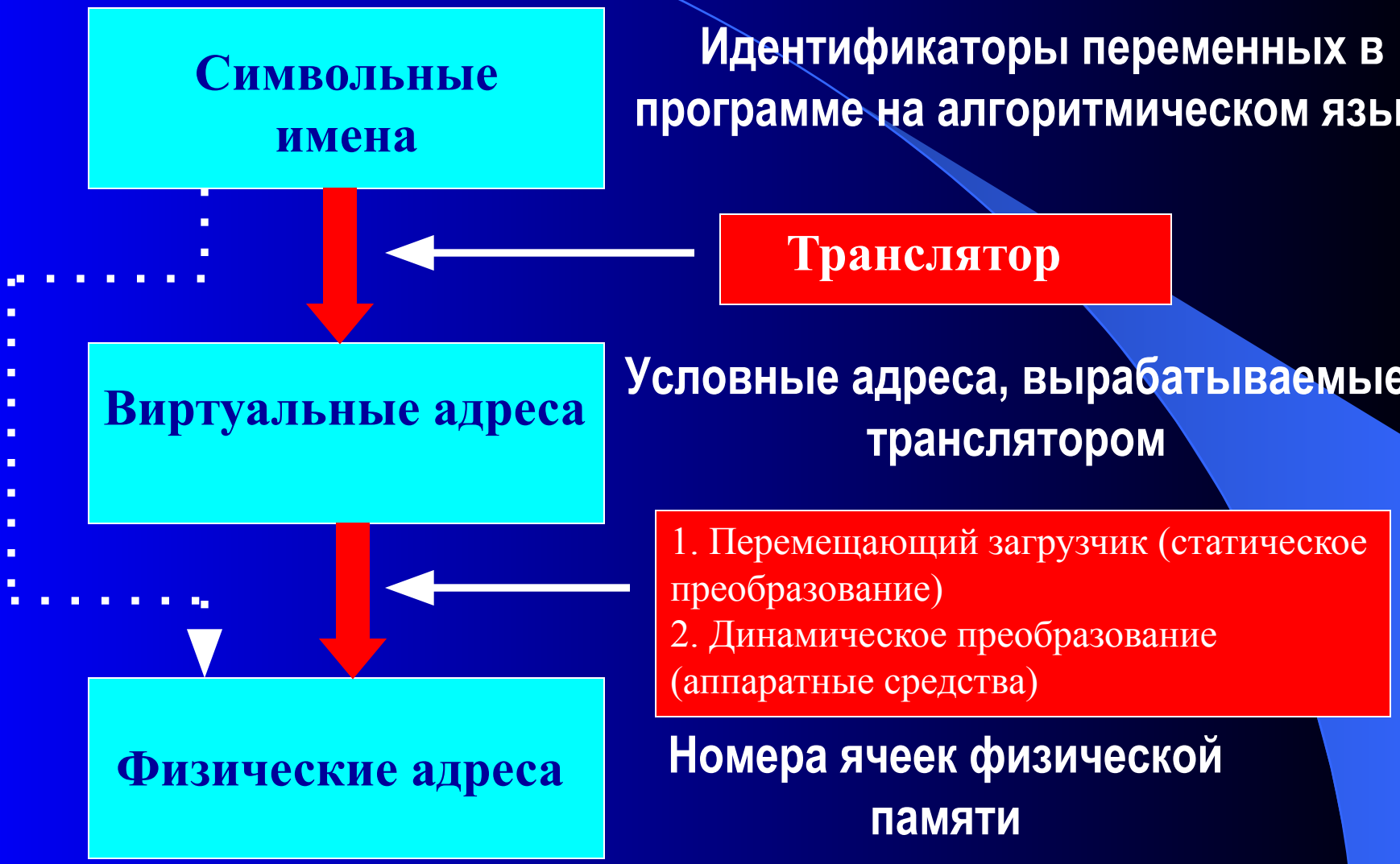
3.2. Функции операционной системы по управлению памятью

Распределение памяти в однопрограммных ОС

Функции операционной системы по управлению памятью в мультипрограммных системах

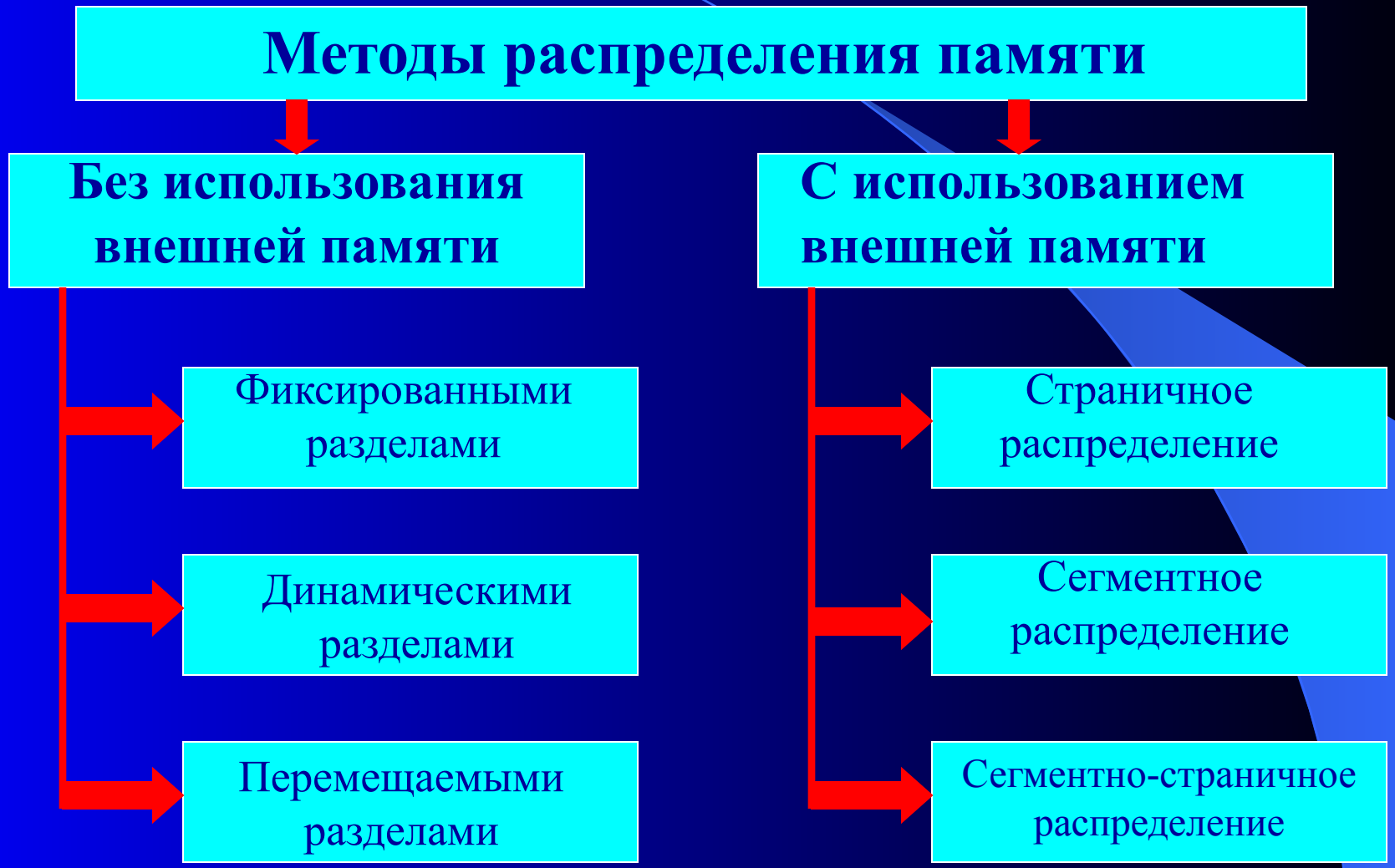
- отслеживание (учет) свободной и занятой памяти;
- первоначальное и динамическое распределение памяти процесса приложений и сомой ОС;
- освобождение памяти при завершении процессов;
- настройка адресов программы на конкретную область физической памяти;
- полное или частичное вытеснение кодов и данных 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.

Типы адресов

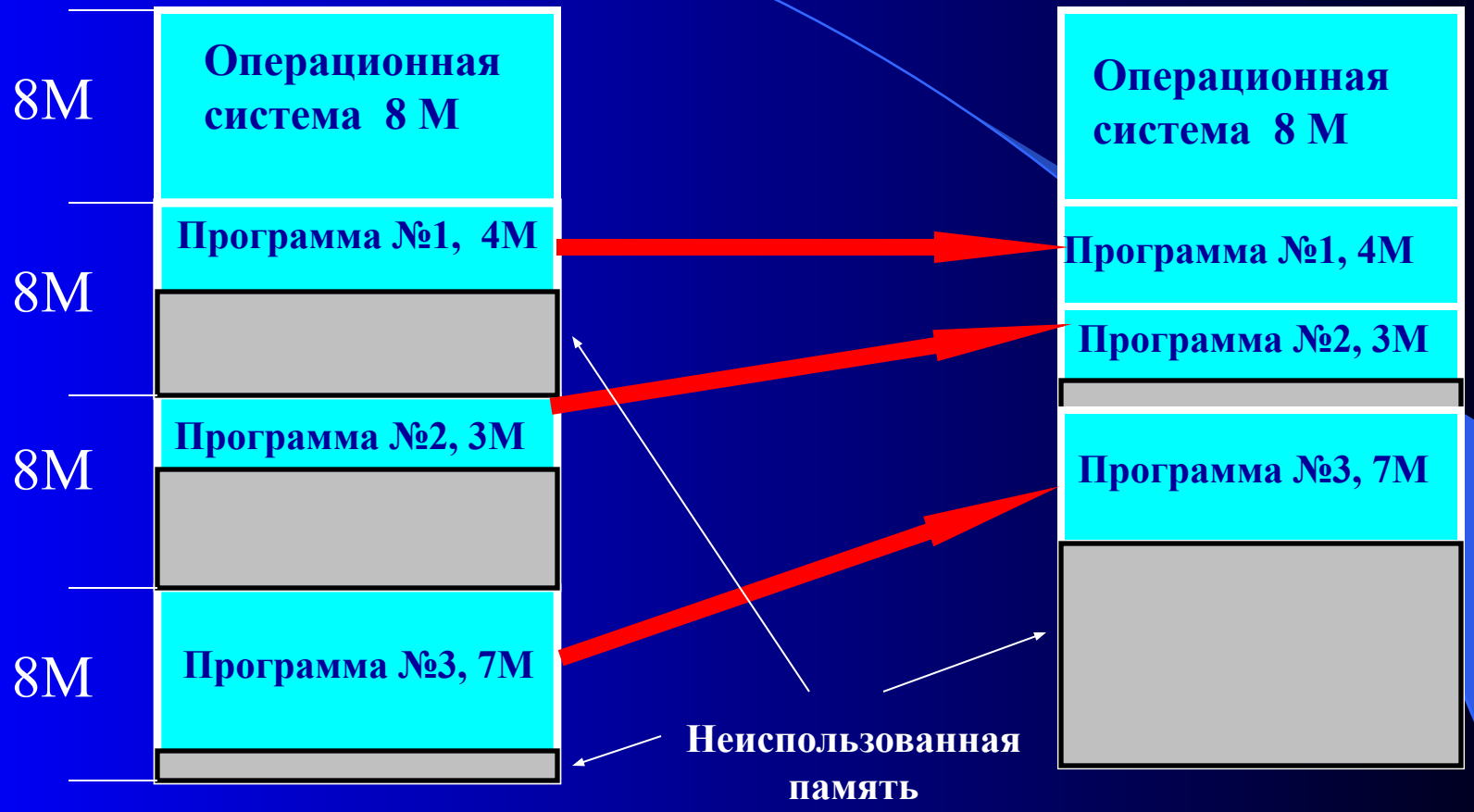


3.3. Алгоритмы распределение памяти

3.3.1. Классификация методов распределения памяти

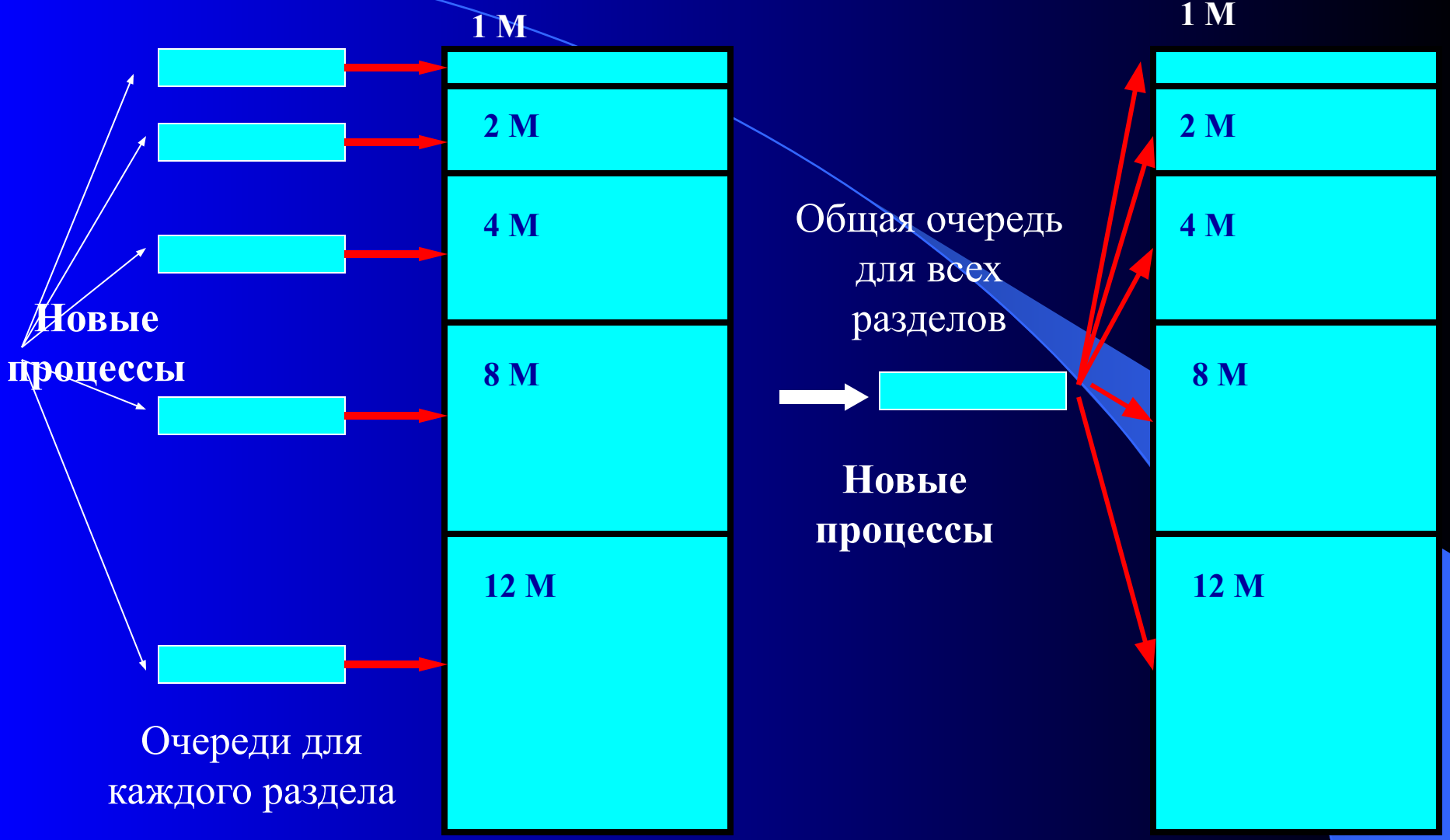


3.3.2. Распределение памяти фиксированными разделами (MFT в OS/360)



Разделы одинакового размера

Разделы разного размера


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.

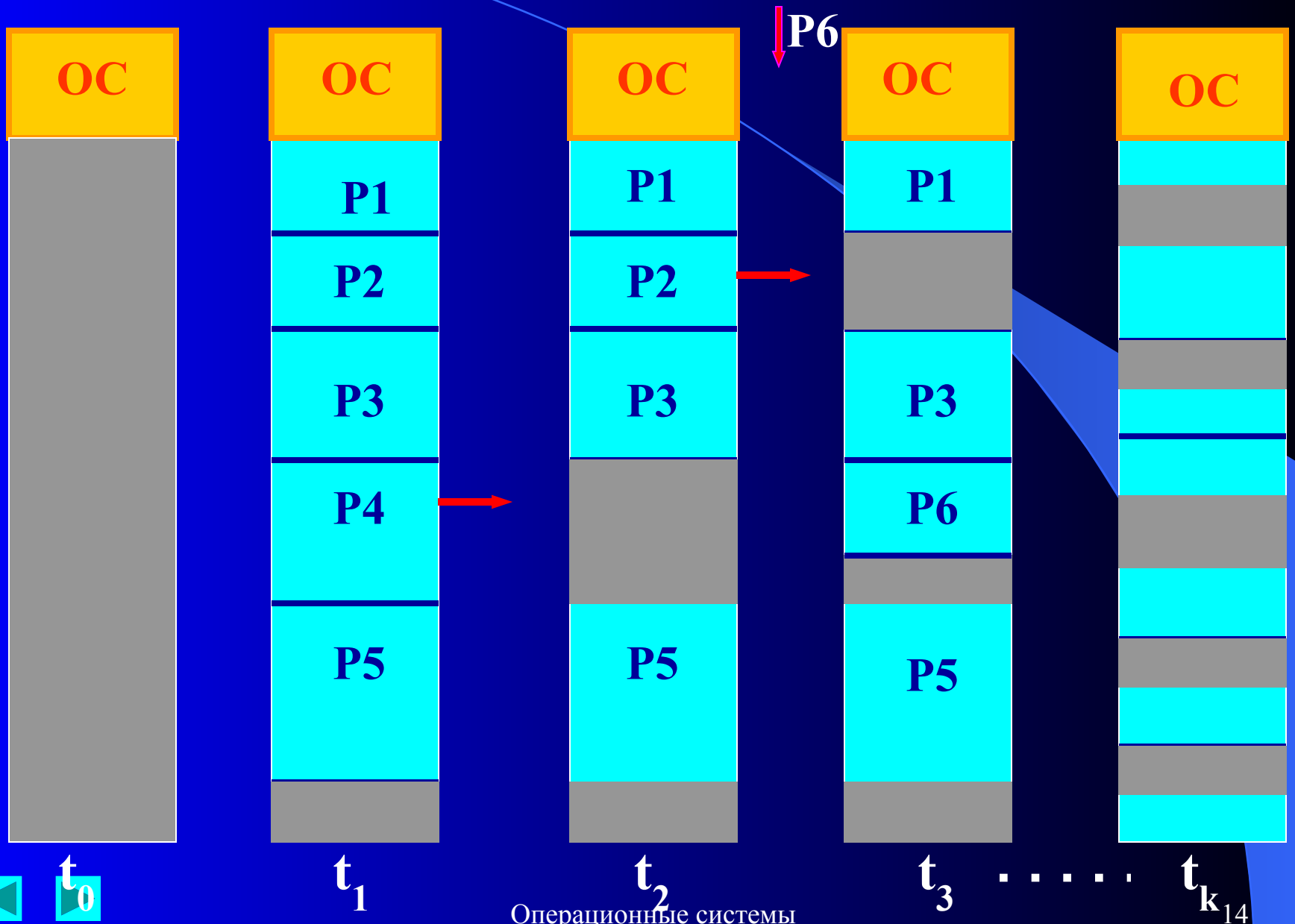
Недостаток: возможно неэффективное использование памяти за счет «простоя» больших разделов при наличии только небольших процессов.

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.

3.3.3. Распределение памяти динамическими разделами



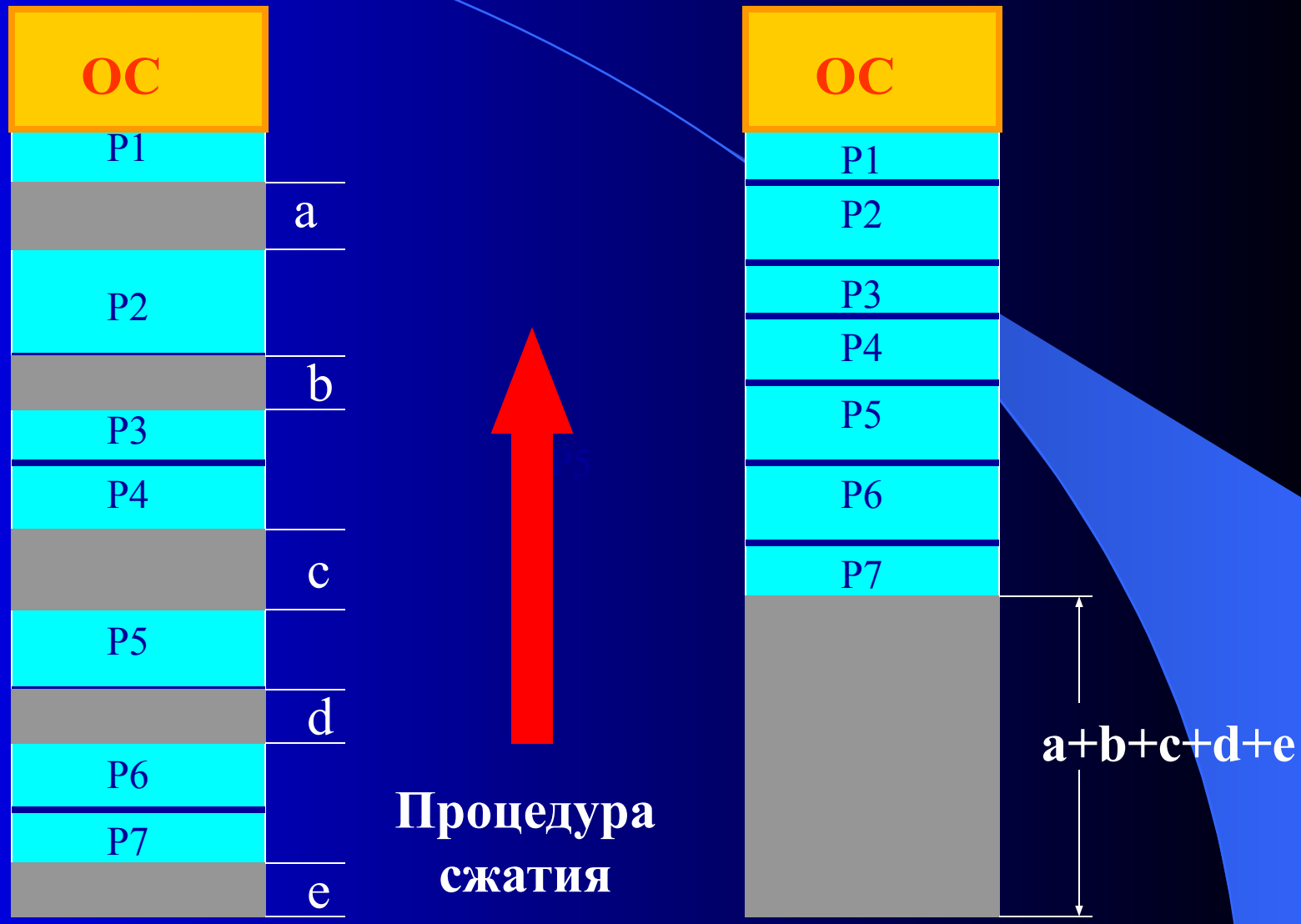
Распределение памяти динамическими разделами

Достоинства: большая гибкость по сравнению с фиксированными разделами. **Недостаток:** 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.

3.3.4. 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

3.4. Виртуальная память

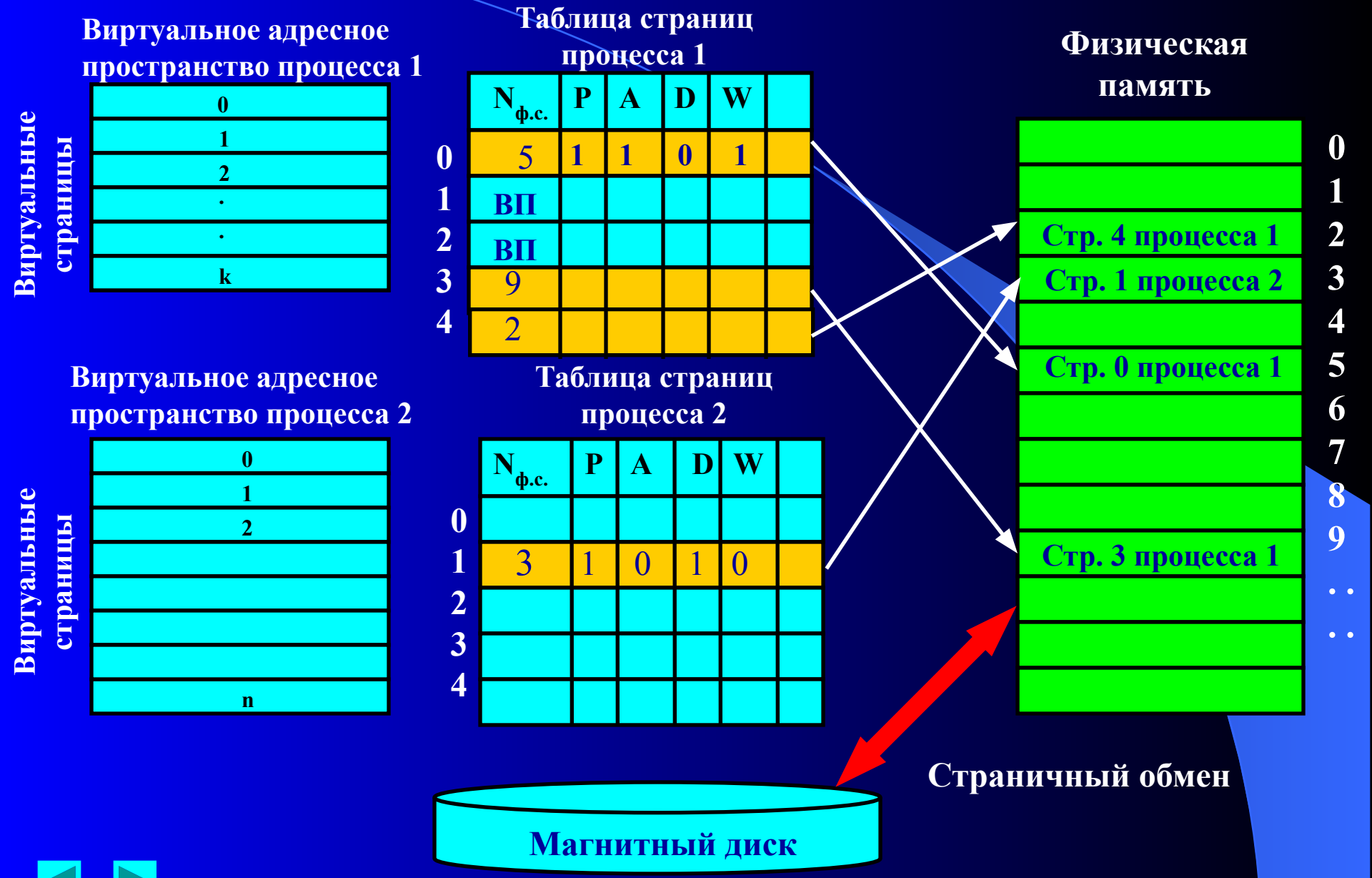
3.4.1. Методы структуризации виртуального адресного пространства

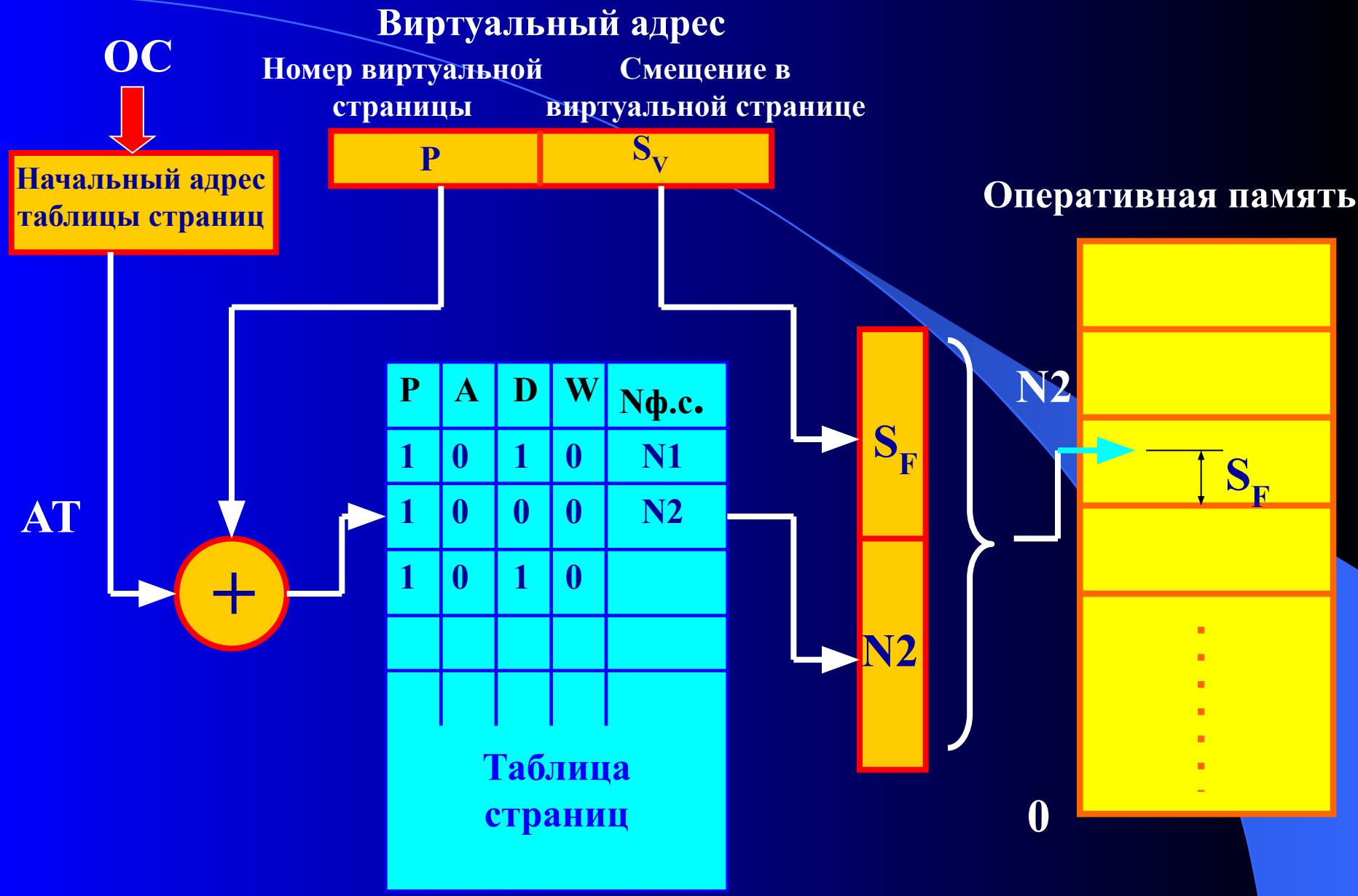
1962 г. – Kilburn T. и др. “One –Level Storage System”

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).**

3.4.2. Страничная организация виртуальной памяти



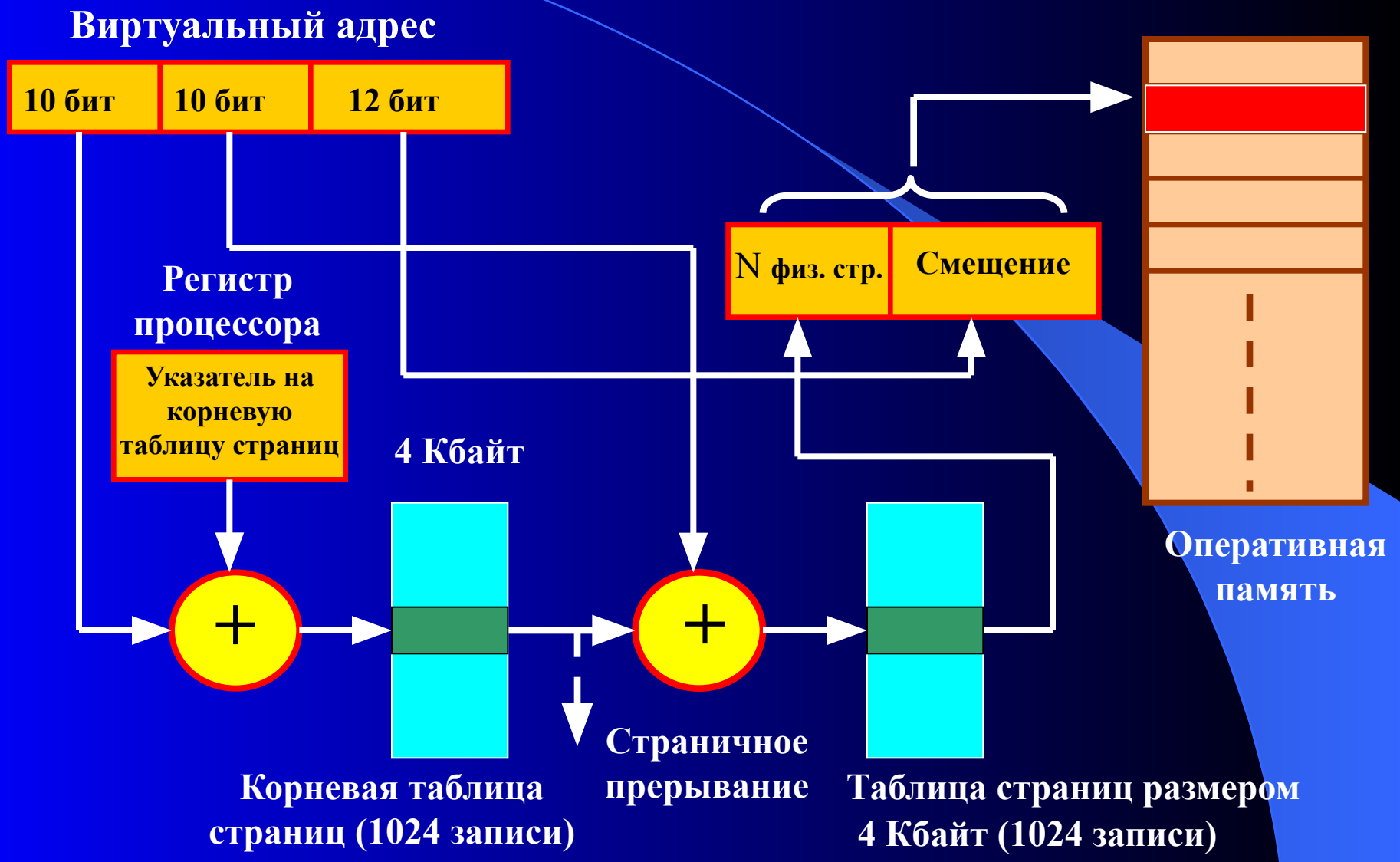

Виртуальный адрес
ОС
Номер виртуальной страницы
Смещение в виртуальной странице
P
S_V
Начальный адрес таблицы страниц
Оперативная память
AT
+
P A D W Nф.с.
1 0 1 0 N1
1 0 0 0 N2
1 0 1 0
Таблица страниц
S_F
N2
N2
S_F
0

3.4.3. Оптимизация функционирования страничной виртуальной памяти

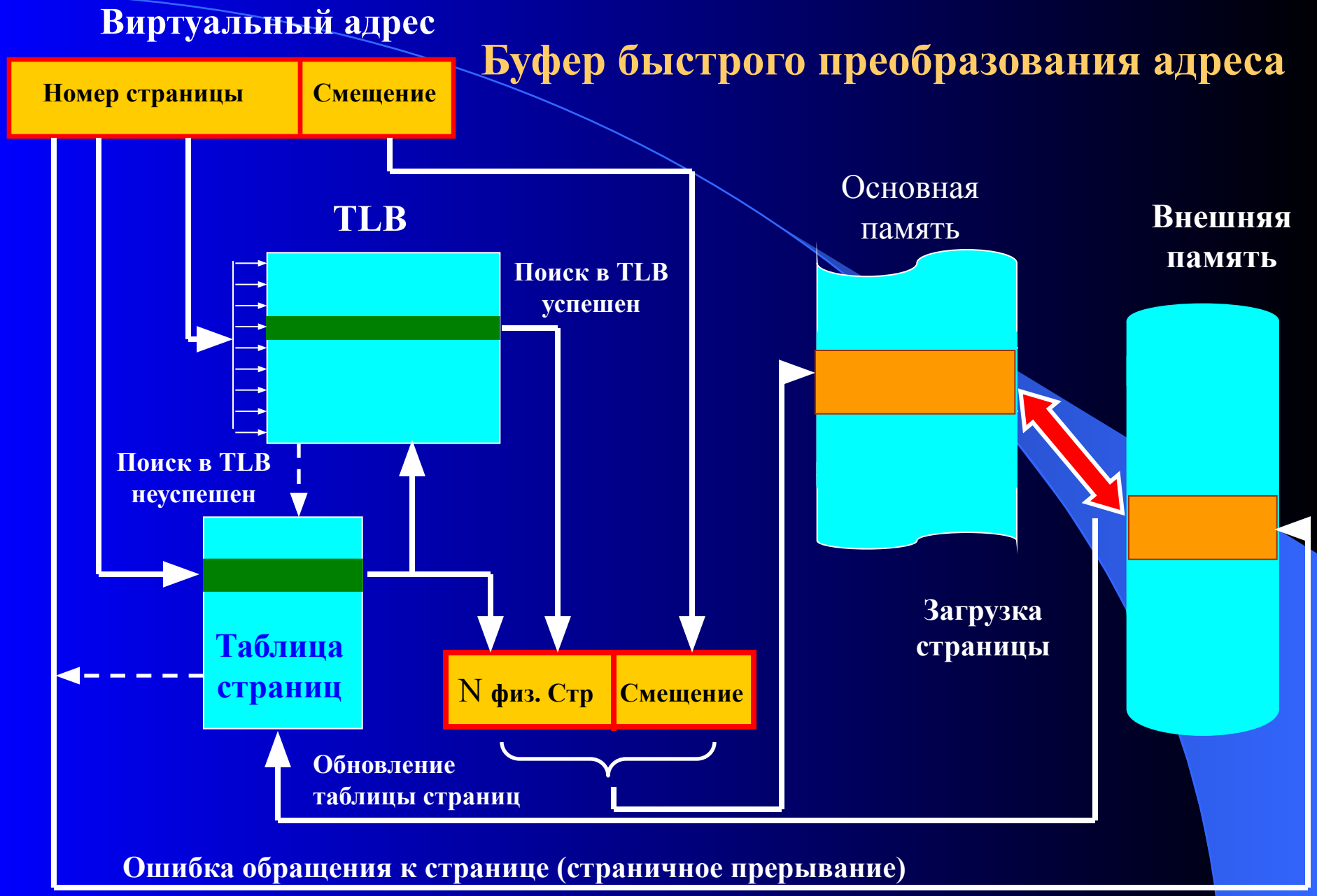
Методы повышения эффективности функционирования страничной виртуальной памяти:

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.

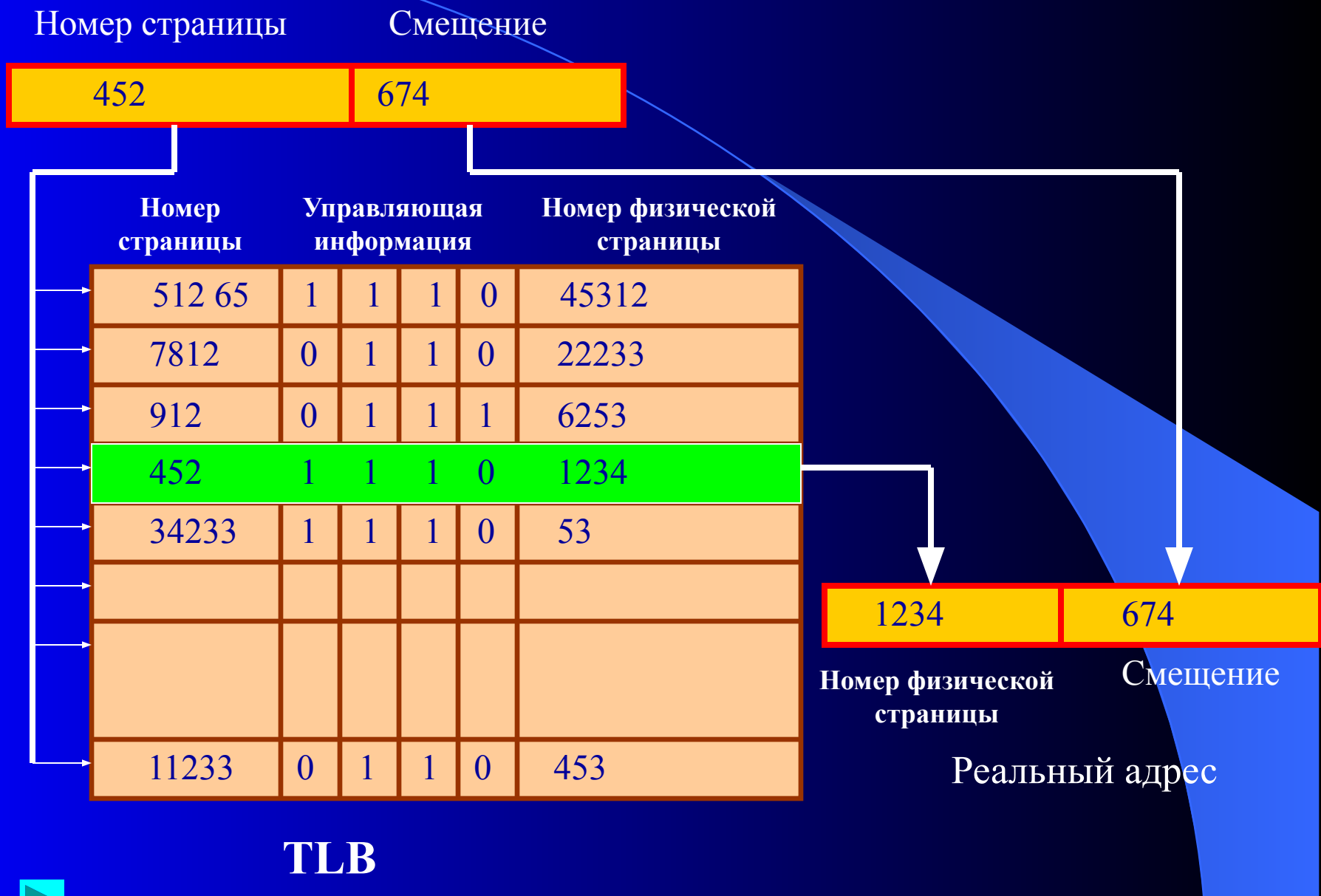
Двухуровневая страничная организация

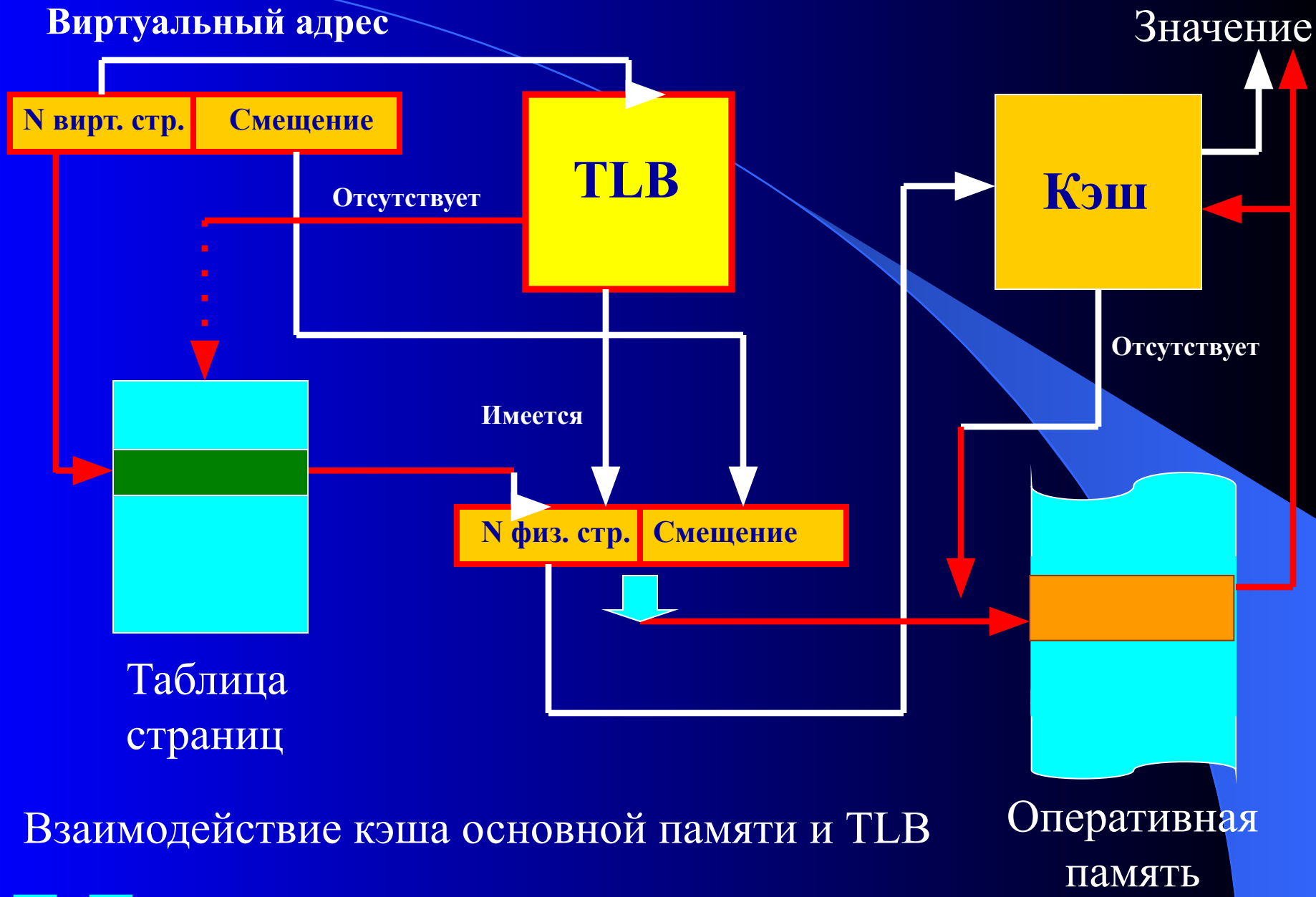


Буфер быстрого преобразования адреса



Ассоциативное отображение

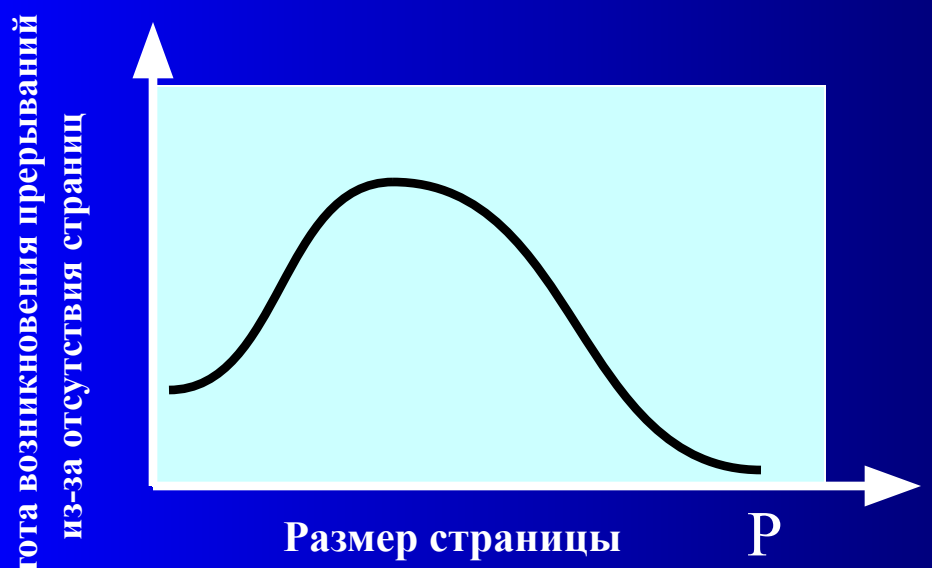




Взаимодействие кэша основной памяти и TLB

Оптимальный размер страниц

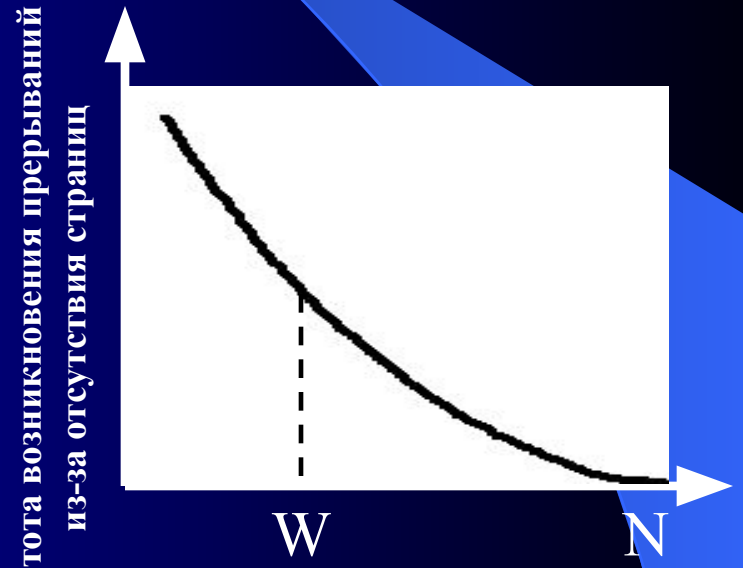
1. С уменьшением размера страницы уменьшается внутренняя фрагментация.
2. С уменьшением размера страницы увеличивается объем страничных таблиц и следовательно накладные расходы на работу виртуальной памяти.
3. С увеличением размера страниц повышается скорость работы диска.
4. Частота страничных прерываний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



P – размер процесса в страницах

N – общее количество страниц процесса

W – размер рабочего множества



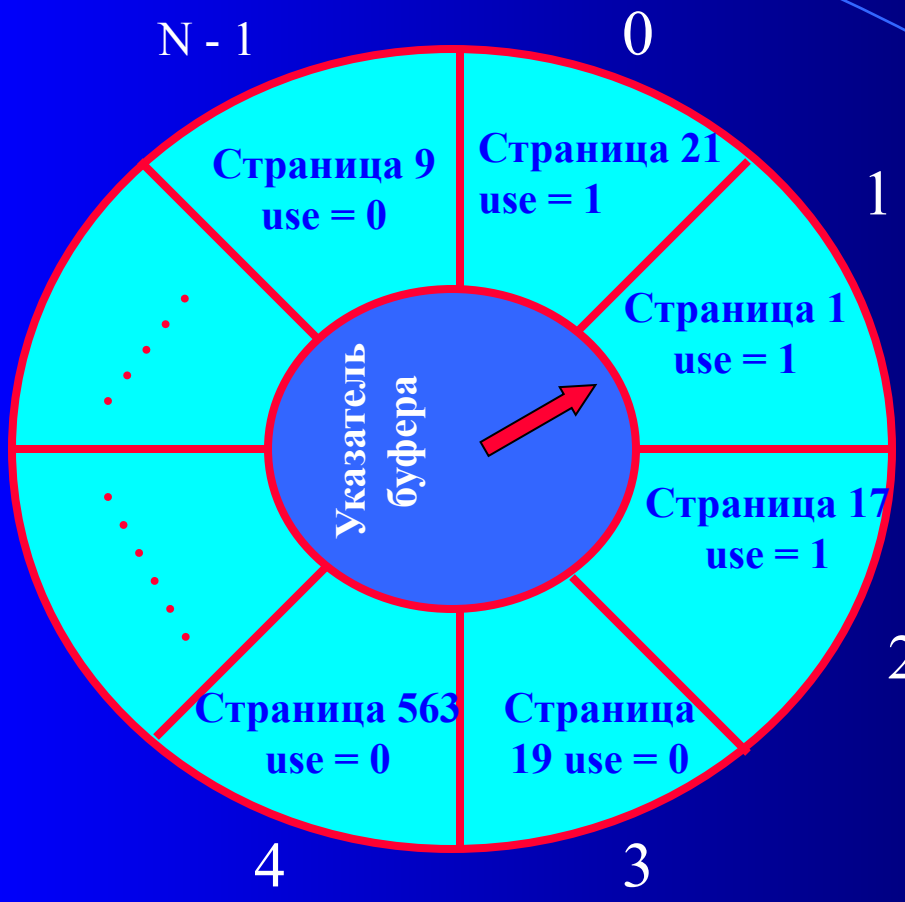
Управление страничным обменом

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.**

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

3.4.4. Сегментная организация виртуальной памяти

При компиляции возможно создание следующих сегментов:

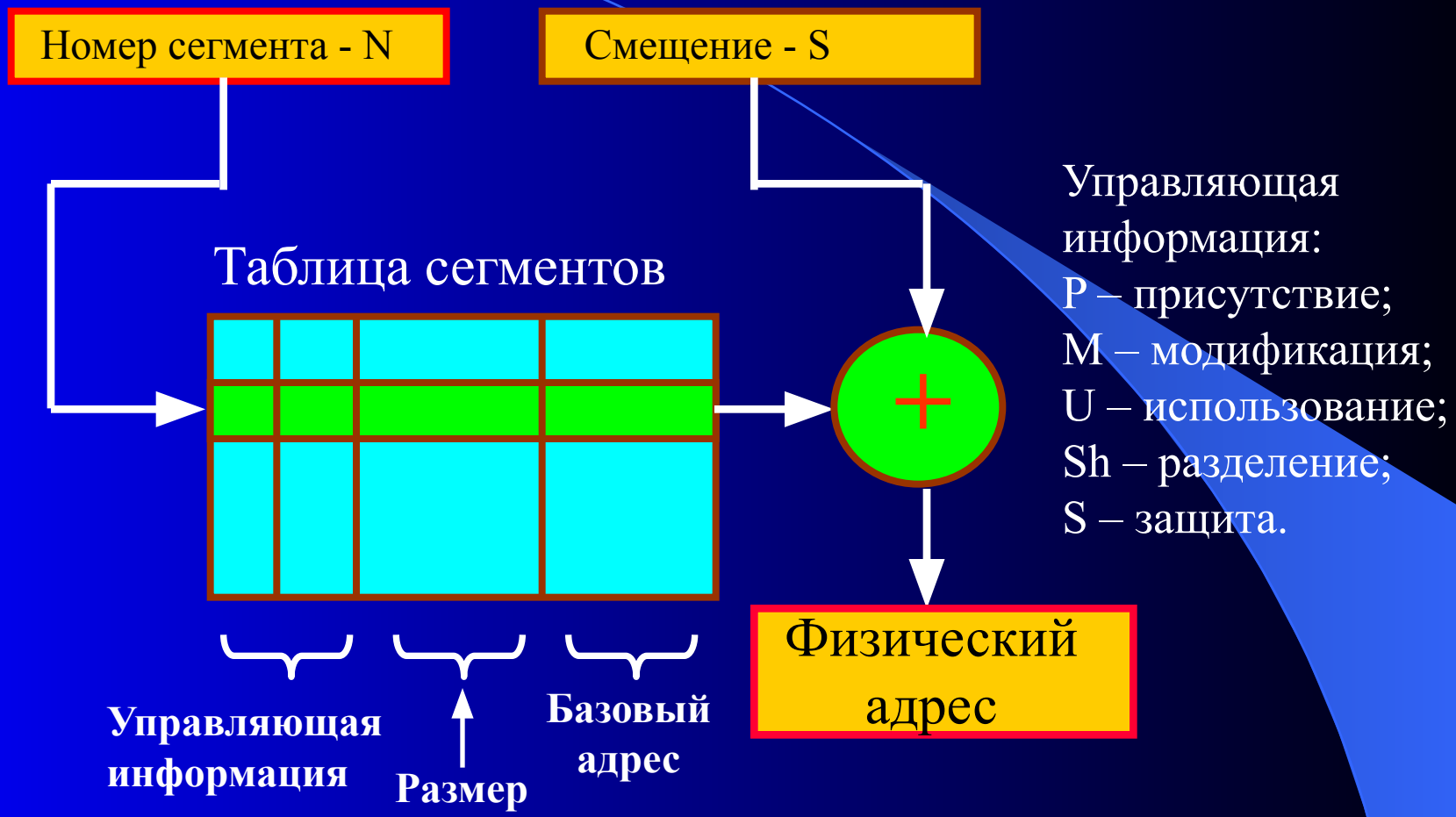
1. Исходный текст, сохраненный для печати листинга программы.
2. Символьная таблица, содержащая имена и атрибуты переменных.
3. Таблица констант.
4. Дерево грамматического разбора, содержащее синтаксический анализ программы.
5. Стек, используемый для процедурных вызовов внутри компилятора.

Таблица кодировки символов достигла таблицы с исходным текстом

Сравнение страничной и сегментной организации памяти

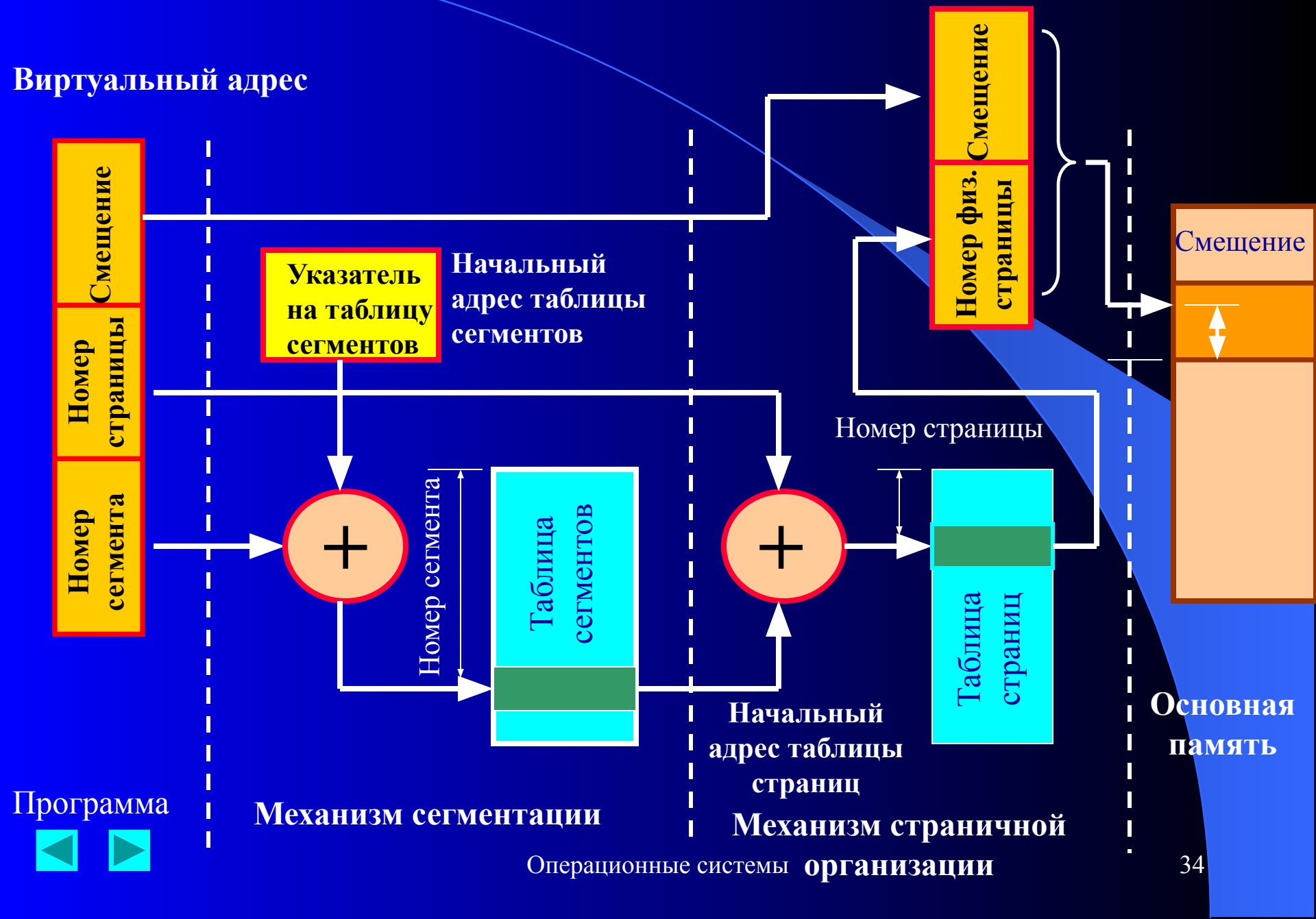
Вопрос	Страничная	Сегментация
Нужно ли программисту знать о том, что используется эта техника?	Нет	Да
Сколько в системе линейных адресных пространств?	1	Много
Может ли суммарное адресное пространство превышать размеры физической памяти?	Да	Да
Возможно ли разделение процедур и данных, а также раздельная 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

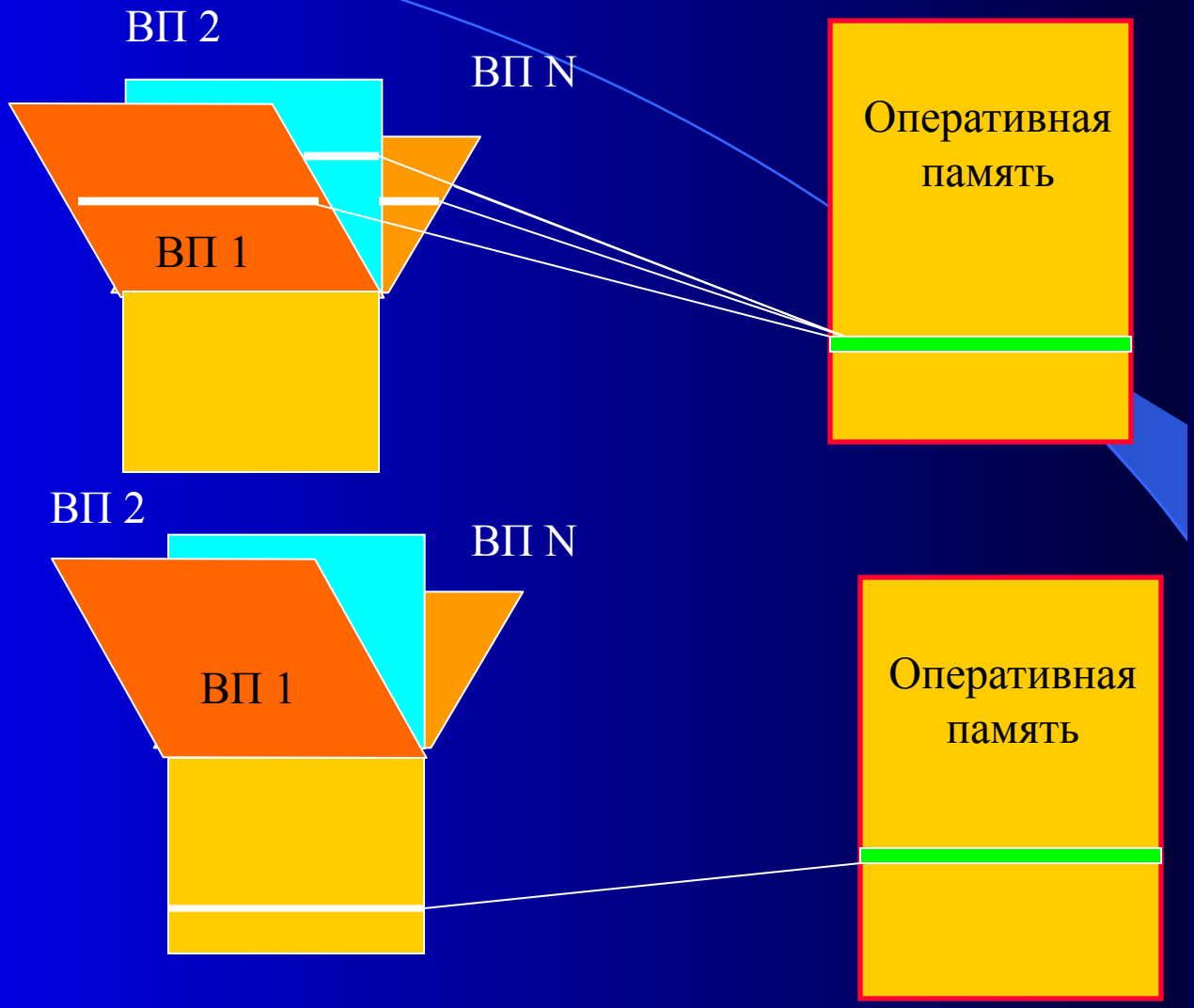
Виртуальный адрес



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

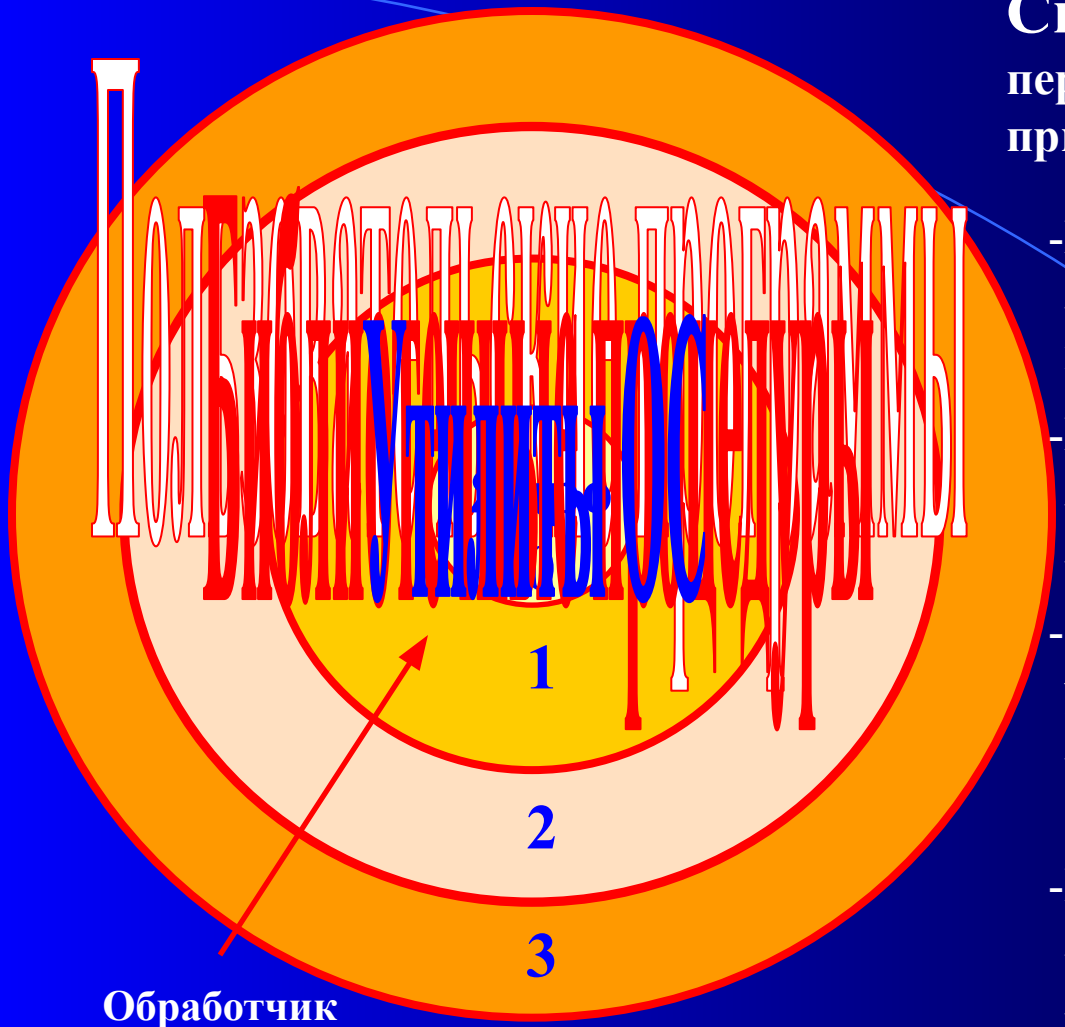
Сегментно-страничная организация виртуальной памяти





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).



Система защиты использует переменные, характеризующие уровень привилегий:

-DPL (Descriptor Privilege Level) – задается полем DPL в дескрипторе сегмента;

-RPL (Requested Privilege Level) – запрашиваемый уровень привилегий, задается полем RPL селектора сегмента;

-CPL (Current Privilege Level) – текущий уровень привилегий выполняемого кода задается полем RPL селектора кодового сегмента (фиксируется в PSW);

-EPR (Effective Privilege Level) – эффективный уровень привилегий запроса.

Контроль доступа к сегменту данных осуществляется , если EPL <= DPL, где EPL = max { CPL, RPL }. Значение RPL – уровня запрашиваемых привилегий – определяется полем RPL селектора, указывающего на запрашиваемый сегмент.