# *Лекция 3.* Основные подсистемы ОС: подсистема управления процессами и потоками

# Основные функции подсистемы управления процессами

- создание и уничтожение процессов (т.е. структур данных, связанных с процессами)
- поддержание очередей заявок процессов на ресурсы
- защита ресурсов, выделенных данному процессу, от остальных процессов организовывать совместное использование ресурсов
- обеспечивать прерывание и возобновление некоторого процесса
- функции синхронизации процессов, позволяющие процессу приостанавливать свое выполнение до наступления какого-либо события в системе
- предоставить средства межпроцессного взаимодействия

# Определение процесса

- Процесс – задача в стадии выполнения в системе.
- Процесс – заявка на потребление всех видов ресурсов (в системах, где определено понятие «поток», кроме одного ресурса - процессорного времени).
- Поток – средство распараллеливания вычислений внутри процесса.

# Преимущества ввода понятия поток

- Создание потоков требует от ОС меньше накладных расходов, чем процессов.
- Мультипрограммирование на уровне потоков более эффективно (распараллеливание).
- Использование потоков приводит к созданию более структурированных и понятных программ.

# Описание процесса в системе

- идентификатор процесса
- идентификатор пользователя, создавшего процесс
- данные о расположении в памяти исполняемого модуля
- степень привилегированности процесса (приоритет и права доступа)

# Планирование потоков

Работа по определению того, в какой момент необходимо прервать выполнение текущего активного потока и какому потоку предоставить возможность выполняться, называется ***планированием.*** Планирование по сути есть стратегия

# При планировании потоков учитываются

- приоритет потоков
- время их ожидания в очереди
- накопленное время выполнения
- интенсивность обращений к вводу-выводу
- другие факторы

# Планирование потоков состоит в решении задач:

- определение момента времени для смены текущего активного потока
- выбор для выполнения потока из очереди готовых потоков

# Диспетчеризация

***Диспетчеризация*** заключается в реализации найденного в результате планирования (динамического или статистического) решения, то есть в переключении процессора с одного потока на другой

Диспетчеризация – это тактика действий

# Диспетчеризация процессов (потоков)

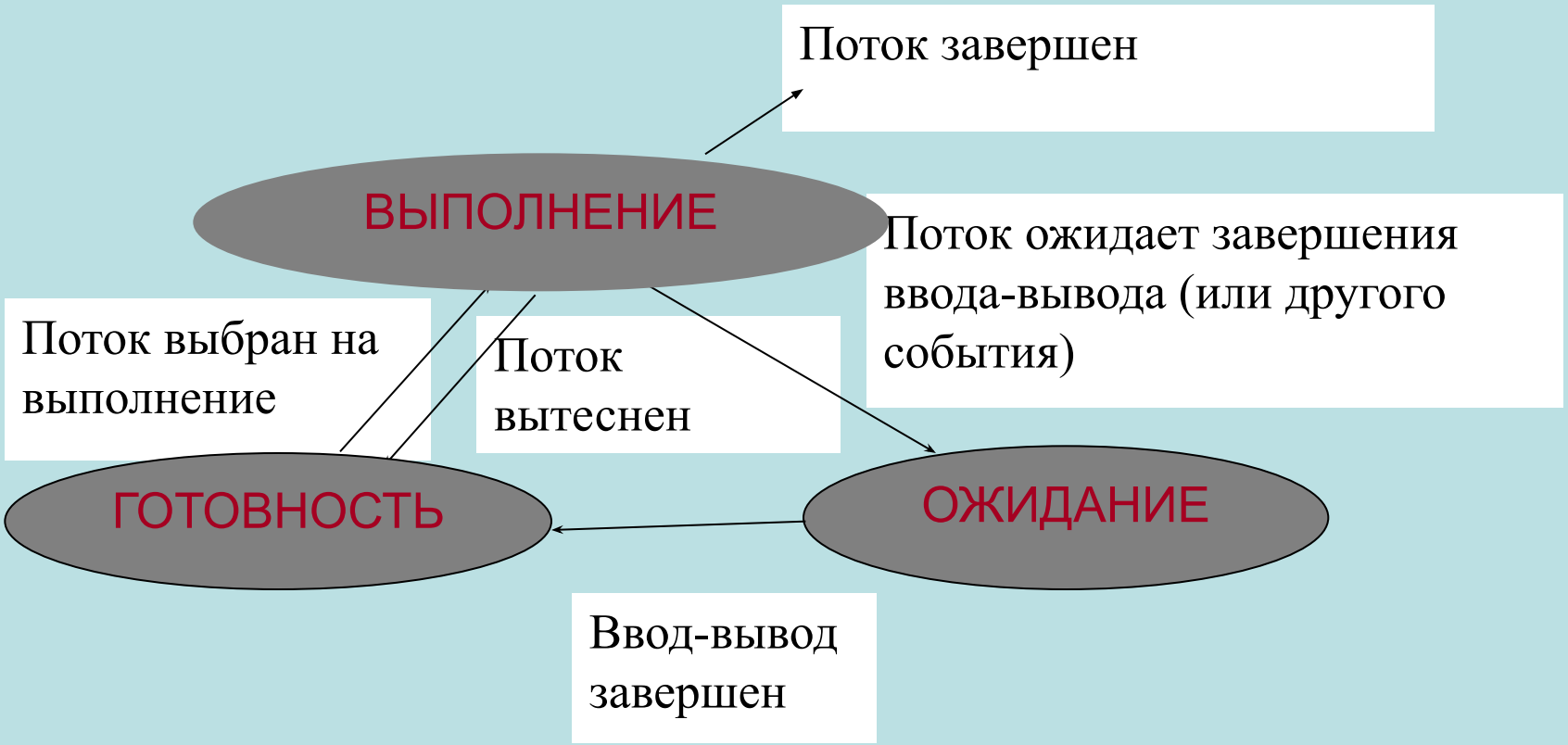
Диспетчеризация сводится к следующему:

- сохранение контекста текущего потока, который требуется сменить;
- загрузка контекста нового потока, выбранного в результате планирования;
- запуск нового потока на выполнение.

# Состояния потока

- *выполнение* - активное состояние потока, во время которого поток обладает всеми необходимыми ресурсами и непосредственно выполняется процессором;
- *ожидание* – пассивное состояние потока, находясь в котором, поток заблокирован по своим внутренним причинам (ждет осуществления некоторого события, например, завершения операции ввода-вывода, получения сообщения от другого потока или освобождения какого-либо необходимого ему ресурса);
- *готовность* – также пассивное состояние потока, но в этом случае поток заблокирован в связи с внешним по отношению к нему обстоятельством (имеет все требуемые для него ресурсы, готов выполняться, однако процессор занят выполнением другого потока).

# Диаграмма смены состояний потоков



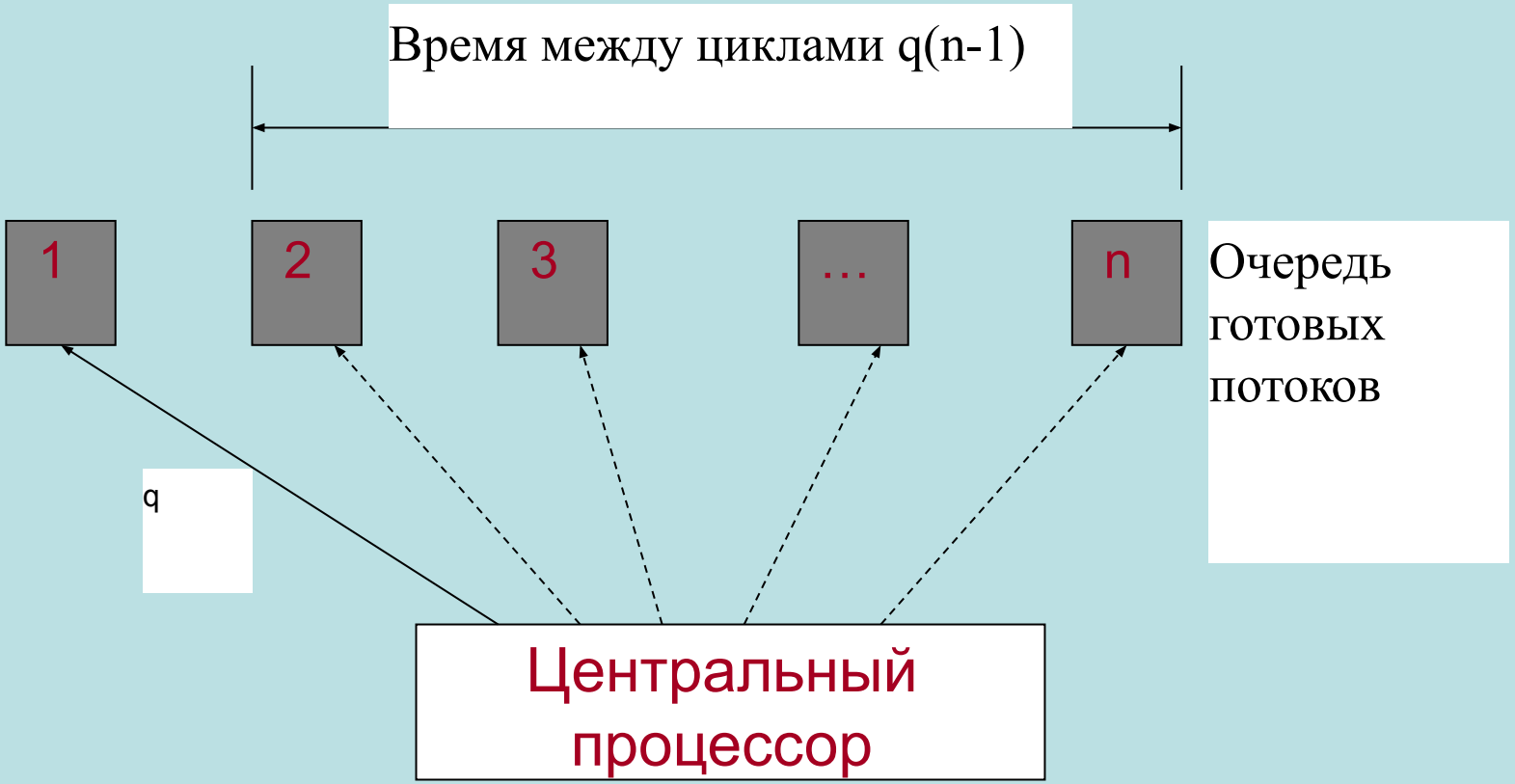
# Алгоритмы планирования выполнения потоков

- *Невытесняющие (non-preemptive)* алгоритмы основаны на том, что активному потоку позволяется выполняться, пока он сам, по собственной инициативе, не отдаст управление операционной системе для того, чтобы та выбрала из очереди другой готовый к выполнению поток (децентрализованное планирование).
- *Вытесняющие (preemptive)* алгоритмы – это такие способы планирования потоков, в которых решение о переключении процессора с выполнения одного потока на выполнение другого потока принимается операционной системой, а не активной задачей (централизованное планирование).

# Алгоритмы планирования, основанные на квантовании

- Квант – это ограниченный непрерывный период процессорного времени, который предоставляется поочередно каждому потоку для выполнения.
- Смена активного потока происходит, если:
  - поток завершился и покинул систему;
  - произошла ошибка;
  - поток перешел в состояние ожидания;
  - исчерпан квант процессорного времени, отведенный данному потоку.
- Кванты, выделяемые одному потоку, могут быть фиксированной величины, а могут и изменяться в разные периоды жизни потока

# Иллюстрация квантового механизма планирования



# Алгоритмы планирования, основанные на приоритетах

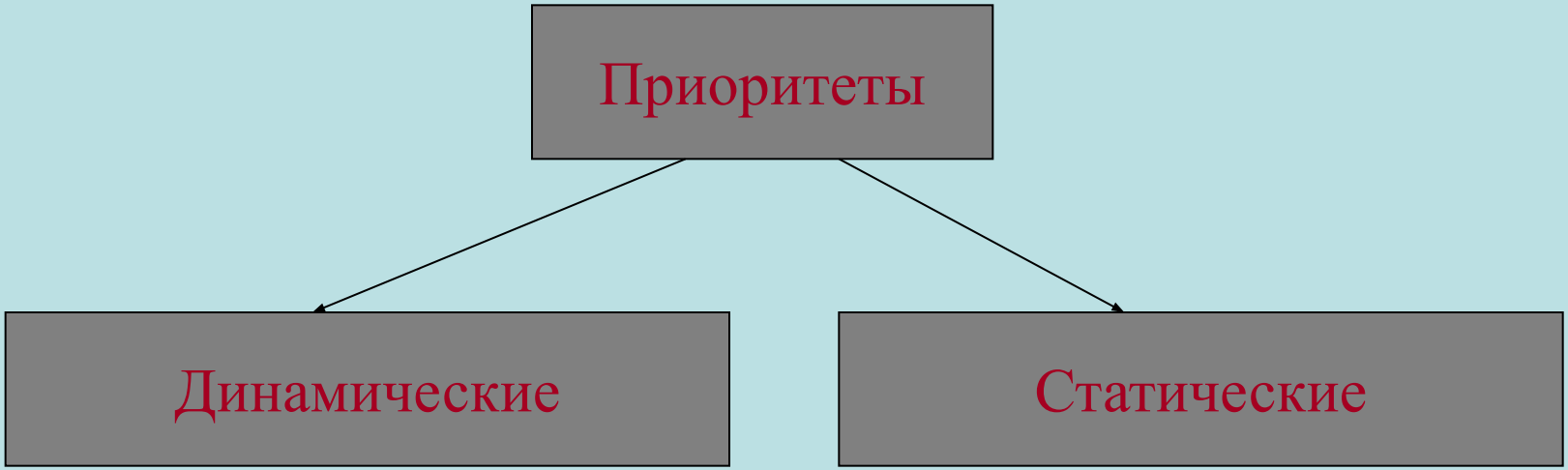
- **Приоритет** - это число, характеризующее степень привилегированности потока при использовании ресурсов вычислительной машины, в частности - процессорного времени: чем выше приоритет, тем выше привилегии, тем меньше времени будет проводить поток в очередях
- Приоритет может выражаться целым или дробным, положительным или отрицательным значением. В некоторых ОС принято, что приоритет потока тем выше, чем больше (в арифметическом смысле) число, обозначающее приоритет. В других системах, наоборот, чем меньше число, тем выше приоритет.
- В большинстве операционных систем, поддерживающих потоки, приоритет потока непосредственно связан с приоритетом процесса, в рамках которого выполняется данный поток.

# Назначение приоритетов

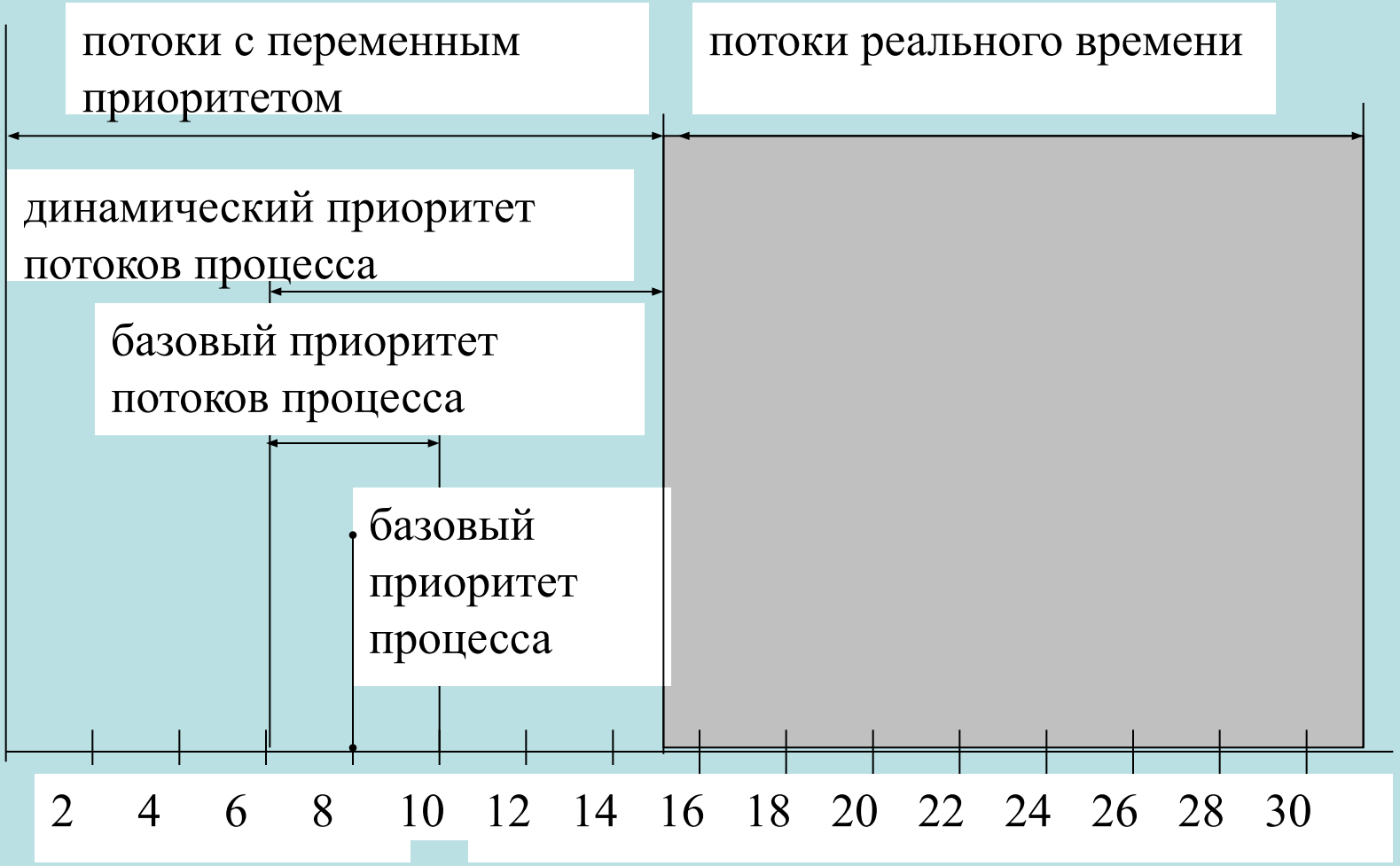
При назначении приоритета вновь созданному процессу ОС учитывает:

- является этот процесс системным или прикладным,
- каков статус пользователя, запустившего процесс,
- было ли явное указание пользователя на присвоение процессу определенного уровня приоритета.

# Разновидности приоритетов

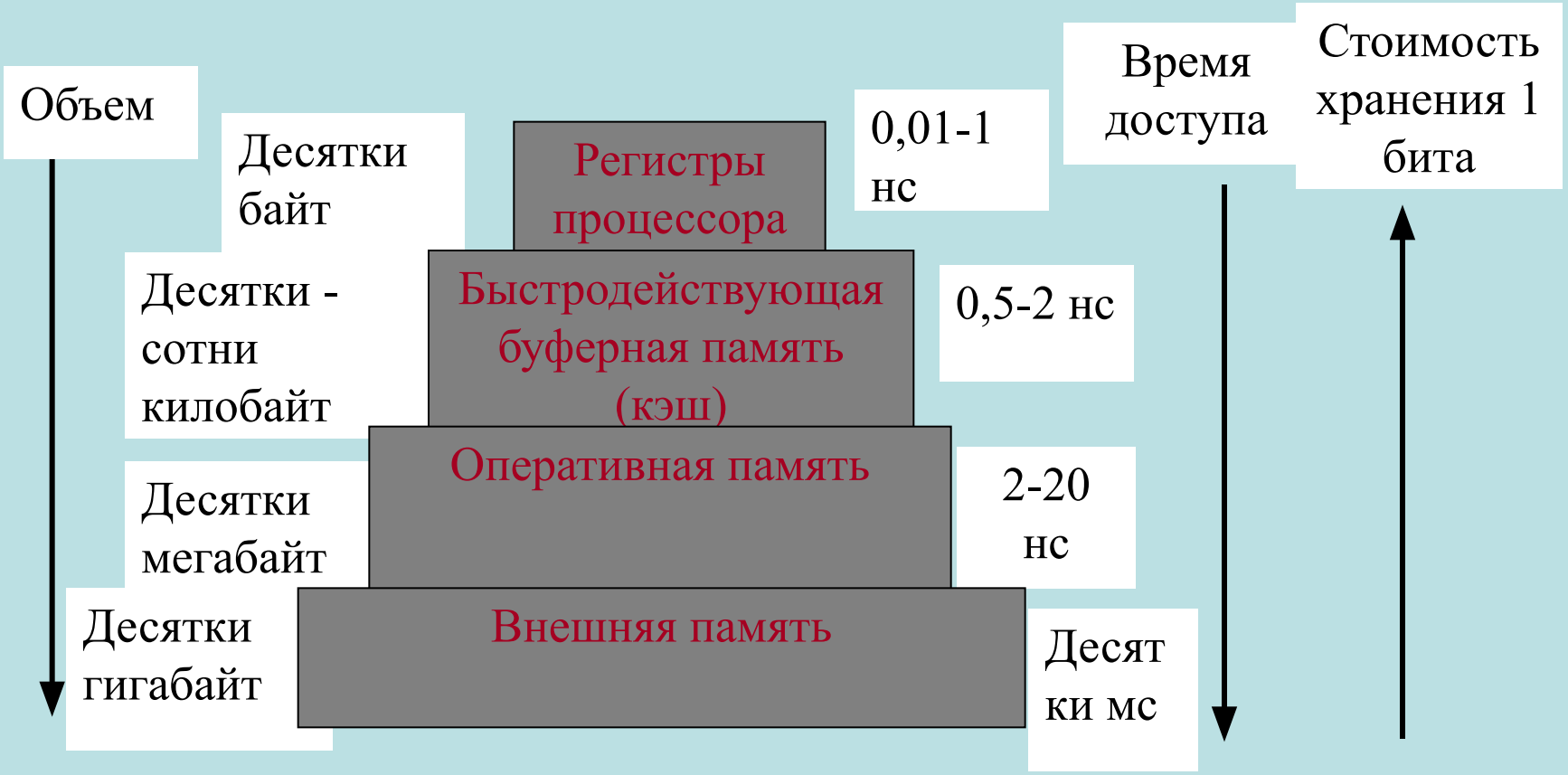


# Схема назначения приоритетов в Windows NT



# Основные подсистемы ОС: подсистема управления памятью

# Иерархия типов памяти



# Регистры процессора

- Имеют разное предназначение: от хранения адреса текущей исполняемой инструкции до хранения и обработки данных боле общего характера.
- Регистры процессора работают со скоростью самого процессора, в противном случае они представляли бы узкое место для компьютера в целом.
- Количество регистров процессора (и их использование) напрямую зависит от архитектуры самого процессора. Изменить число регистров процессора невозможно, если только не перейти на процессор с другой архитектурой. Поэтому число регистров процессора можно считать константой.

# Кэш

- Подсистема кэша в конструкции современных компьютеров может быть многоуровневой
- Уровни кэша часто нумеруются, при этом, чем меньше номер, тем ближе он к процессору. На многих компьютерах имеется два уровня кэша:
  - кэш L1, который обычно находится непосредственно внутри процессора и работает со скоростью процессора
  - кэш L2 обычно является частью процессорного модуля, его скорость равна (или почти равна) скорости процессора, при этом он немного больше и медленнее кэша L1.
- Некоторые компьютеры (чаще всего это высокопроизводительные серверы) имеют ещё кэш L3, который обычно расположен на материнской плате. Как можно догадаться, кэша L3 больше (и вероятнее всего медленнее), чем L2.

# Основная память

- Называется также оперативной памятью, или оперативным запоминающим устройством (ОЗУ)
- На самом низком уровне это микросхемы памяти — микросхемы, которые собственно «запоминают» информацию. Эти микросхемы соединяются с внешним миром четырьмя типами контактов:
  - Контакты питания (чтобы микросхема могла работать)
  - Контакты данных (чтобы микросхема могла обмениваться данными)
  - Контакты чтения/записи (определяющие, выполняется ли чтение или запись данных)
  - Контакты адреса (определяющие, куда записываются или откуда считываются данные)

# Чтение / запись в ОЗУ

- Сохранения данных в ОЗУ:
  - На контактах микросхемы появляются сохраняемые данные.
  - На контактах адреса появляется адрес, по которому будут сохранены данные.
  - На контакте чтение/запись устанавливается режим записи.
- Чтение данных из ОЗУ:
  - На контактах адреса появляется адрес нужных данных.
  - На контакте чтение/запись устанавливается режим чтения.
  - Затребованные данные считываются с контактов данных.

# Функции подсистемы управления основной памятью

- отслеживание свободной и занятой памяти;
- выделение памяти процессам и освобождение памяти по завершении процессов;
- вытеснение кодов и данных из оперативной памяти на диск (полное или частичное), когда размеры основной памяти не достаточны для размещения в ней всех процессов, и возвращение их в оперативную память, когда в ней освобождается место;
- настройка адресов процесса на конкретную область физической памяти;
- защита памяти

# Организация основной памяти

Под **организацией памяти** обычно понимают то, каким образом представляется и используется основная память.

- организация разделов (существует ли разбиение на какие-то части, постоянно ли такое разбиение, одинакового или разного размера эти разделы)
- программа занимает один раздел или несколько
- можно ли программу разбивать на части для занятия разделов, занятых не полностью
- и т.д.

# Стратегии управления памятью

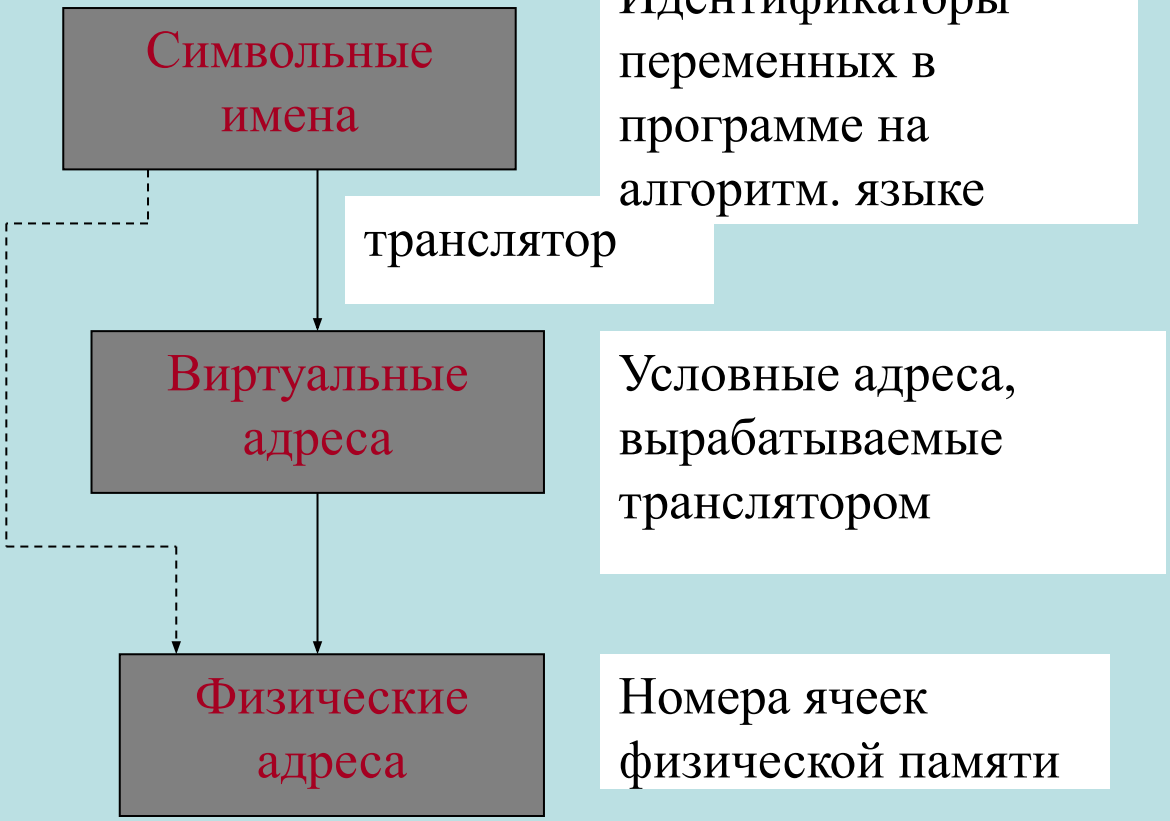
- стратегии выборки (загрузки);
  - а) стратегии выборки по запросу (по требованию);
  - б) стратегии упреждающей выборки;
- стратегии размещения;
- стратегии замещения.

*Стратегии выборки* ставят своей целью определять, когда следует «втолкнуть» очередной блок программы или данных в основную память.

Стратегии *размещения* ставят своей целью определить, в какое место основной памяти следует помещать поступающую программу.

Стратегии *замещения* ставят своей целью определить, какой блок программы или данных следует вывести («вытолкнуть») из основной памяти, чтобы освободить место для записи поступающих программ или данных.

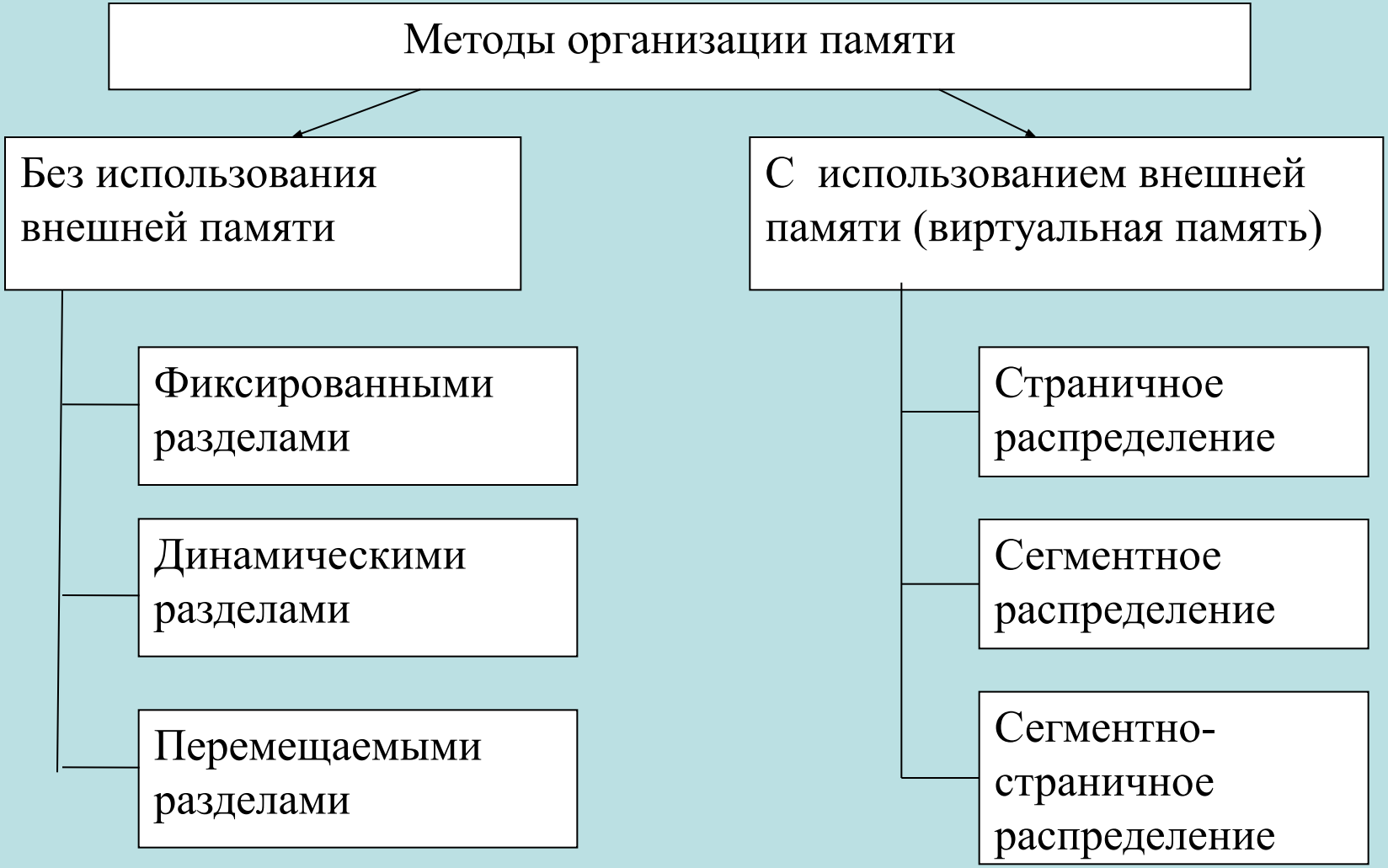
# Типы адресов



# Типы адресов

- **Символьные имена** присваивает пользователь при написании программы на алгоритмическом языке или ассемблере.
- **Виртуальные адреса**, называемые иногда *математическими,* или *логическими адресами,* вырабатывает транслятор, переводящий программу на машинный язык. Поскольку во время трансляции в общем случае не известно, в какое место оперативной памяти будет загружена программа, то транслятор присваивает переменным и командам виртуальные (условные) адреса, обычно считая по умолчанию, что начальным адресом программы будет нулевой адрес.
- **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**.

# Организация памяти



# Распределение памяти фиксированными разделами

Самым простым способом управления оперативной памятью является разделение ее на несколько разделов фиксированной величины. Это может быть выполнено вручную оператором во время старта системы или во время ее генерации.

Функции подсистемы при работе с фиксированными разделами

- Сравнить объем памяти, требуемый для вновь поступившего процесса, с размерами свободных разделов и выбрать подходящий раздел.
- Осуществить загрузку программы в один из разделов и настройку адресов.

# Достоинства и недостатки работы с жесткими разделами

- Достоинство – простота реализации
- Недостаток – жесткость (уровень мультипрограммирования ограничен числом разделов)

# Распределение памяти динамическими разделами

- 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.

# Функции подсистемы при работе с динамическими разделами

- 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.
- После завершения процесса корректировка таблиц свободных и занятых областей.

# Достоинства и недостатки работы с динамическими разделами

- Достоинство – гибкость (по сравнению с фиксированными разделами)
- Недостаток – фрагментация памяти.

  **Фрагментация** – это наличие большого числа несмежных участков свободной памяти очень маленького размера (фрагментов).

# Понятие виртуальной памяти

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

**Виртуальное адресное пространство** — это максимально доступное приложению адресное пространство. Объём виртуального адресного пространства зависит от архитектуры компьютера и операционной системы.

**Виртуальная память** — схема адресации памяти, при которой память представляется программному обеспечению непрерывной и однородной, в то время как в реальности для фактического хранения данных используются отдельные (разрывные) области различных видов памяти, включая и долговременную память (жёсткие диски, твёрдотельные накопители).

# Основные задачи, решаемые подсистемой виртуальной памяти

- 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.

# 2 подхода к виртуализации памяти

- *свопинг (swapping), или подкачка,* – 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.

# Достоинства и недостатки свопинга

- *Достоинство* – относительная простота реализации
- *Недостаток* – избыточность откачиваемой информации (когда ОС решает активизировать процесс, для его выполнения, как правило, не требуется загружать в оперативную память все его сегменты полностью - достаточно загрузить небольшую часть кодового сегмента с подлежащей выполнению инструкцией и частью сегментов данных, с которыми работает эта инструкция, а также отвести место под сегмент стека)

# Реализации виртуальной памяти

- *Страничная виртуальная память* организует перемещение данных между памятью и диском страницами – частями виртуального адресного пространства, фиксированного и сравнительно небольшого размера.
- *Сегментная виртуальная память* предусматривает *перемещение* данных сегментами – частями виртуального адресного пространства произвольного размера, полученными с учетом смыслового значения данных.
- *Сегментно-страничная виртуальная память* использует двухуровневое деление: виртуальное адресное пространство делится на сегменты, а затем сегменты делятся на страницы. Единицей перемещения данных здесь является страница. Этот способ управления памятью объединяет в себе элементы обоих предыдущих подходов.

Виртуальное адресное пространство процесса 1

| |
|---|
| 0 |
| 1 |
| 2 |
| 3 |
| 4 |

Таблица страниц процесса 1

| | №ф.с. | Упр. инф. |
|---|---|---|
| 0 | 5 | |
| 1 | ВП | |
| 2 | ВП | |
| 3 | 2 | |
| 4 | 109 | |

Физическая память

| | |
|---|---|
| 0 | |
| 1 | |
| 2 | Стр4, пр1 |
| 3 | |
| 4 | |
| 5 | Стр.0, пр1 |
| 6 | |
| 7 | |
| 8 | Стр 0, пр2 |
| 9 | Стр.5, пр2 |
| | ... |

Виртуальное адресное пространство процесса 2

| |
|---|
| 0 |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |

Таблица страниц процесса 1

| | №ф.с. | Упр. инф. |
|---|---|---|
| 0 | 8 | |
| 1 | ВП | |
| 2 | ВП | |
| 3 | ВП | |
| 4 | ВП | |
| 5 | 9 | |

Страничный обмен

# Страничное распределение памяти

# Страничный файл (файл подкачки)

Для временного хранения сегментов и страниц на диске отводится специальная область: специальный файл (Windows) или раздел (Linux), называемые файлом (разделом) подкачки.

Употребляется также термин страничный файл, т.к. используется работа со страницами (page file, или paging file).

# Виртуальная страница (определение)

Виртуальное адресное пространство каждого процесса делится на части одинакового, фиксированного для данной системы размера. Такая область называется *виртуальной страницей (virtual page)*.

Вся оперативная память машины также делится на части такого же размера, называемые *физическими страницами* (или страничными фреймами). В процессорах Intel Pentium он равен 4 кбайтам.

# Таблица страниц

Для каждого процесса операционная система создает **таблицу страниц** – информационную структуру, содержащую записи обо всех виртуальных страницах процесса.

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.

# Алгоритм работы виртуальной памяти

- При каждом обращении к памяти выполняется поиск номера виртуальной страницы, содержащей требуемый виртуальный адрес
- По этому номеру определяется нужный элемент таблицы страниц, и из него извлекается описывающая страницу информация.
- Анализируется признак присутствия, и, если данная виртуальная страница находится в оперативной памяти, то выполняется преобразование виртуального адреса в физический,
- Если нужная виртуальная страница в данный момент выгружена на диск, то происходит так называемое *страничное прерывание.* Выполняющийся процесс переводится в состояние ожидания, и активизируется другой процесс из очереди процессов, находящихся в состоянии готовности.
- Параллельно программа обработки страничного прерывания находит на диске требуемую виртуальную страницу и пытается загрузить ее в оперативную память.

# Алгоритм работы виртуальной памяти (продолжение)

- Если в памяти имеется свободная физическая страница, то загрузка выполняется немедленно, если же свободных страниц нет, то на основании принятой в данной системе стратегии замещения страниц решается вопрос о том, какую страницу следует выгрузить из оперативной памяти.
- После того как выбрана страница, которая должна покинуть оперативную память, обнуляется ее бит присутствия и анализируется ее признак модификации.
- Если выталкиваемая страница за время последнего пребывания в оперативной памяти была модифицирована, то ее новая версия должна быть переписана на диск.
- Если нет, то принимается во внимание, что на диске уже имеется предыдущая копия этой виртуальной страницы, и никакой записи на диск не производится. Физическая страница объявляется свободной. Из соображений безопасности в некоторых системах освобождаемая страница обнуляется, с тем, чтобы невозможно было использовать содержимое выгруженной страницы.

# Виртуальный и физический адреса

**Виртуальный адрес - ($p$, $s_v$) -**

где:

- p – порядковый номер виртуальной страницы процесса (нумерация страниц начинается с 0),
- $s_v$ – смещение в пределах виртуальной страницы.

**Физический адрес- ($n$, $s_f$)-**

где :

- n – номер физической страницы,
- $s_f$ – смещение в пределах физической страницы.

**Задача подсистемы виртуальной памяти – отображение ($p$, $s_v$) в ($n$, $s_f$).**

# Базисные свойства страничной виртуальной памяти

- объем страницы выбирается равным степени двойки – $2^k$ (это значит - смещение может быть получено простым отделением k младших разрядов в двоичной записи адреса, а оставшиеся старшие разряды адреса представляют собой двоичную запись номера страницы);
- в пределах страницы непрерывная последовательность виртуальных адресов однозначно отображается в непрерывную последовательность физических адресов ( значит, смещения в виртуальном и физическом адресах $s_v$ и $s_f$ равны между собой).

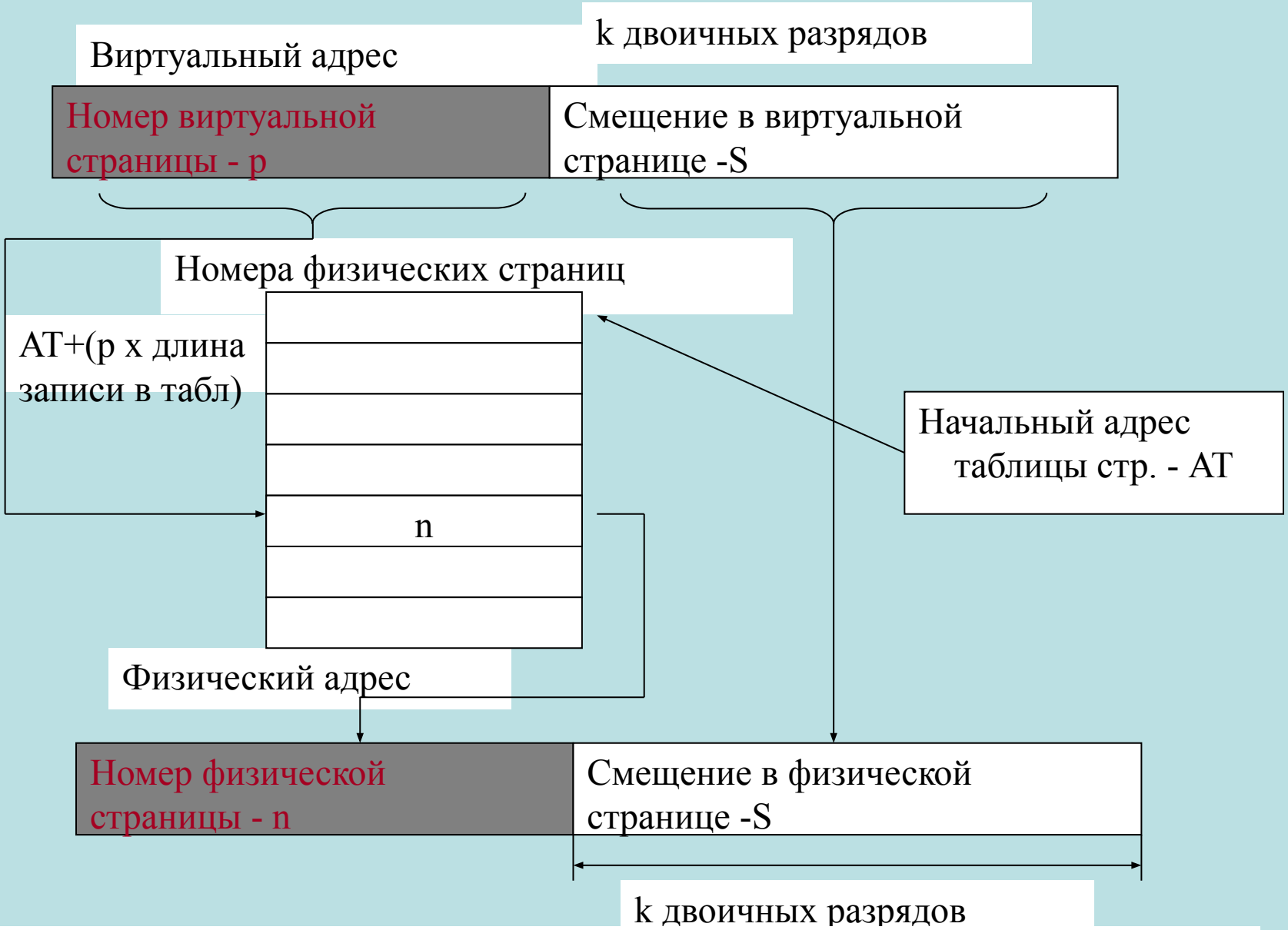



**Схема преобразования виртуального адреса в физический при страничной организации памяти**

# Определение оптимального размера страницы

- Чтобы уменьшить частоту страничных прерываний, следовало бы увеличивать размер страницы.
- Если страница большая:
  - то велик и объем данных, перемещаемых между оперативной и внешней памятью
  - увеличивается и размер фиктивной области в последней виртуальной странице каждого процесса
- Чем меньше страница, тем более объемными являются таблицы страниц процессов и тем больше места они занимают в памяти.
  Вывод: *Типичный размер страницы составляет несколько килобайт, например, наиболее распространенные процессоры x86 и Pentium компании Intel, а также операционные системы, устанавливаемые на этих процессорах, поддерживают страницы размером 4096 байт (4 кбайт).*