


# УПРАВЛЕНИЕ ПАМЯТЬЮ

Память является важнейшим ресурсом, требующим тщательного управления со стороны мультипрограммной операционной системы. Особая роль памяти объясняется тем, что процессор может выполнять инструкции программ только в том случае, если они находятся в памяти. Память распределяется как между модулями прикладных программ, так и между модулями самой операционной системы.



Функциями ОС по управлению памятью в мультипрограммной системе являются:

- отслеживание свободной и занятой памяти;
- выделение памяти процессам и освобождение памяти по завершении процессов;
- вытеснение кодов и данных процессов 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 программы на конкретную область физической памяти;

- дефрагментация памяти;
- защита памяти, которая состоит в том, чтобы не позволить выполняемому процессу записывать или читать данные из памяти, назначенной другому процессу. Эта функция, как правило, реализуется программными модулями ОС в тесном взаимодействии с аппаратными средствами.

## **Типы адресов**

Для идентификации переменных и команд на разных этапах жизненного цикла программы используются символьные имена (метки), виртуальные адреса и физические адреса.




Рисунок 1 - 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.
- 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**. Диапазон возможных адресов виртуального пространства у всех процессов является одним и тем же. Тем не менее каждый процесс имеет собственное виртуальное адресное пространство.

В разных операционных системах используются разные **способы структуризации** виртуального адресного пространства. В одних ОС виртуальное адресное пространство процесса подобно физической памяти представлено в виде непрерывной линейной последовательности виртуальных адресов.



Такую структуру адресного пространства называют **плоской** (flat). При этом виртуальным адресом является единственное число, представляющее собой смещение относительно начала виртуального адресного пространства (Рисунок 2, а). Адрес такого типа называют линейным виртуальным адресом.

В других ОС виртуальное адресное пространство делится на части, называемые сегментами (или секциями, или областями). В этом случае помимо линейного адреса может быть использован виртуальный адрес, представляющий собой пару чисел  $(n, m)$ , где  $n$  определяет сегмент, а  $m$  — смещение внутри сегмента (Рисунок 2, б).

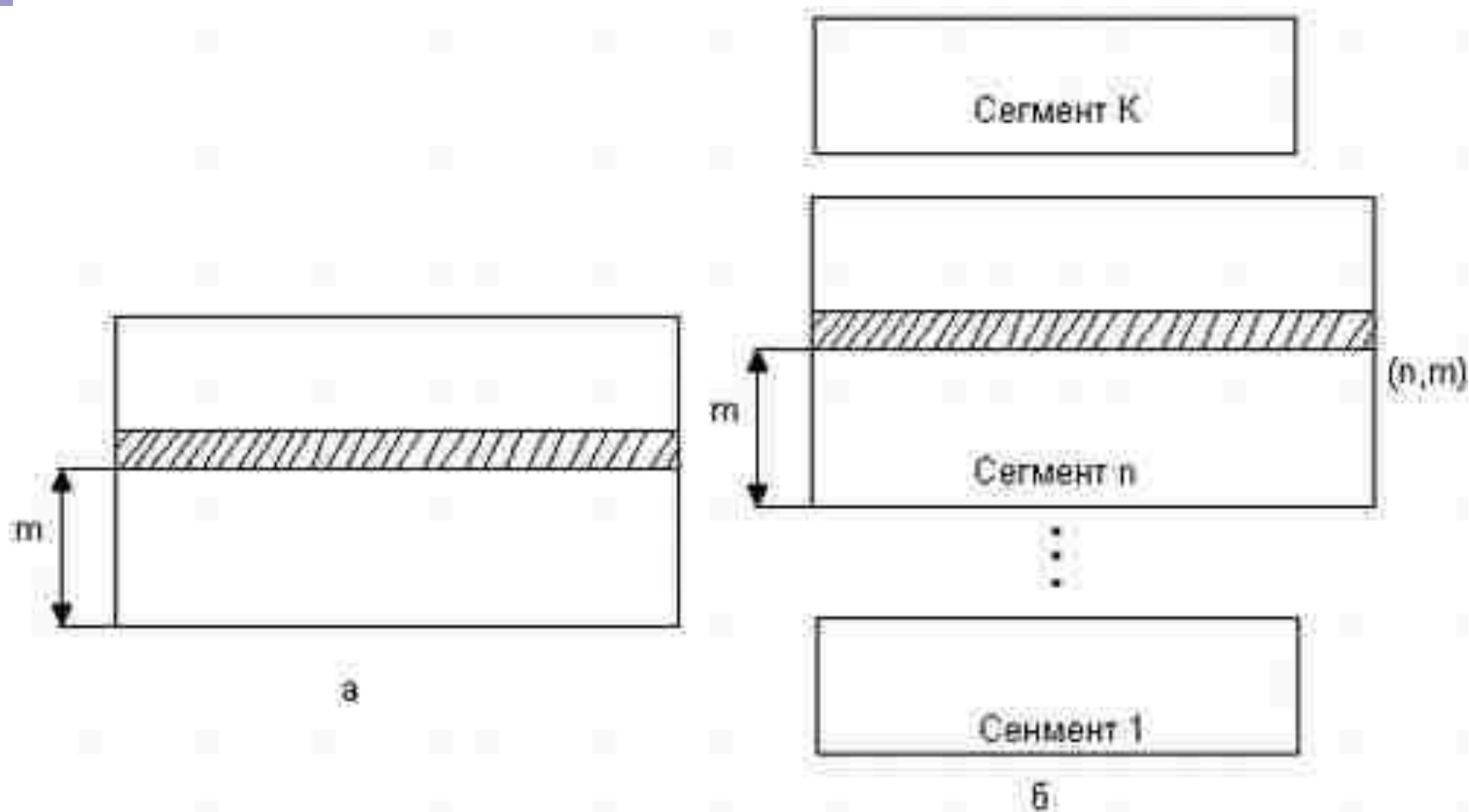



Рисунок 2 - Типы виртуальных адресных пространств: плоское (а), сегментированное (б)




Существуют два принципиально отличающихся подхода к преобразованию виртуальных адресов в физические.

В первом случае замена виртуальных адресов на физические выполняется **один раз для каждого процесса** во время начальной загрузки программы в память. Специальная системная программа — перемещающий загрузчик — на основании имеющихся у нее исходных данных о начальном адресе физической памяти, в которую предстоит загружать программу, а также информации, предоставленной транслятором об адресно-зависимых элементах программы, выполняет загрузку программы, совмещая ее с заменой виртуальных адресов физическими.



Второй способ заключается в том, что программа загружается в память в неизменном виде в виртуальных адресах, то есть операнды инструкций и адреса переходов имеют те значения, которые выработал транслятор.

Во время выполнения программы при каждом обращении к оперативной памяти выполняется преобразование виртуального адреса в физический.



Последний способ является более **гибким**: в то время как перемещающий загрузчик жестко привязывает программу к первоначально выделенному ей участку памяти, динамическое преобразование виртуальных адресов позволяет перемещать программный код процесса в течение всего периода его выполнения. Но использование **перемещающего загрузчика более экономично**, так как в этом случае преобразование каждого виртуального адреса происходит только один раз во время загрузки, а при динамическом преобразовании — при каждом обращении по данному адресу.

Необходимо различать максимально **возможное** виртуальное адресное пространство процесса и **назначенное** (выделенное) процессу виртуальное адресное пространство. В первом случае речь идет о размере виртуального адресного пространства, определяемом архитектурой компьютера, на котором работает ОС, и, в частности, разрядностью его схем адресации (32-битная, 64-битная и т. п.). Например, при работе на компьютерах с 32-разрядными процессорами Intel Pentium операционная система может предоставить каждому процессу виртуальное адресное пространство до 4 Гбайт ( $2^{32}$ ).



Сегодня для машин универсального назначения типична ситуация, когда объем виртуального адресного пространства превышает доступный объем оперативной памяти. В таком случае операционная система для хранения данных виртуального адресного пространства процесса, не помещающихся в оперативную память, использует внешнюю память, которая в современных компьютерах представлена жесткими дисками. Именно на этом принципе основана **виртуальная память** — наиболее совершенный механизм, используемый в операционных системах для управления памятью.

Виртуальное адресное пространство и виртуальная память — это различные механизмы и они не обязательно реализуются в операционной системе одновременно.

Обычно виртуальное адресное пространство процесса делится на две непрерывные части: **системную и пользовательскую**. В некоторых ОС (например, Windows NT) эти части имеют одинаковый размер — по 2 Гбайт.

Часть виртуального адресного пространства каждого процесса, отводимая под сегменты ОС, является идентичной для всех процессов. Поэтому при смене активного процесса заменяется только вторая часть виртуального адресного пространства, содержащая его индивидуальные сегменты, как правило, — коды и данные прикладной программы.

# Алгоритмы распределения памяти



Рисунок 3 - Классификация методов распределения памяти

## Распределение памяти фиксированными разделами.

Простейший способ управления оперативной памятью состоит в том, что память разбивается на несколько областей **фиксированной величины**, называемых **разделами**. Такое разбиение может быть выполнено вручную оператором во время установки системы. После этого границы разделов не изменяются.

Очередной новый процесс, поступивший на выполнение, помещается либо в общую очередь (Рисунок 4, а), либо в очередь к некоторому разделу (Рисунок 4, б).



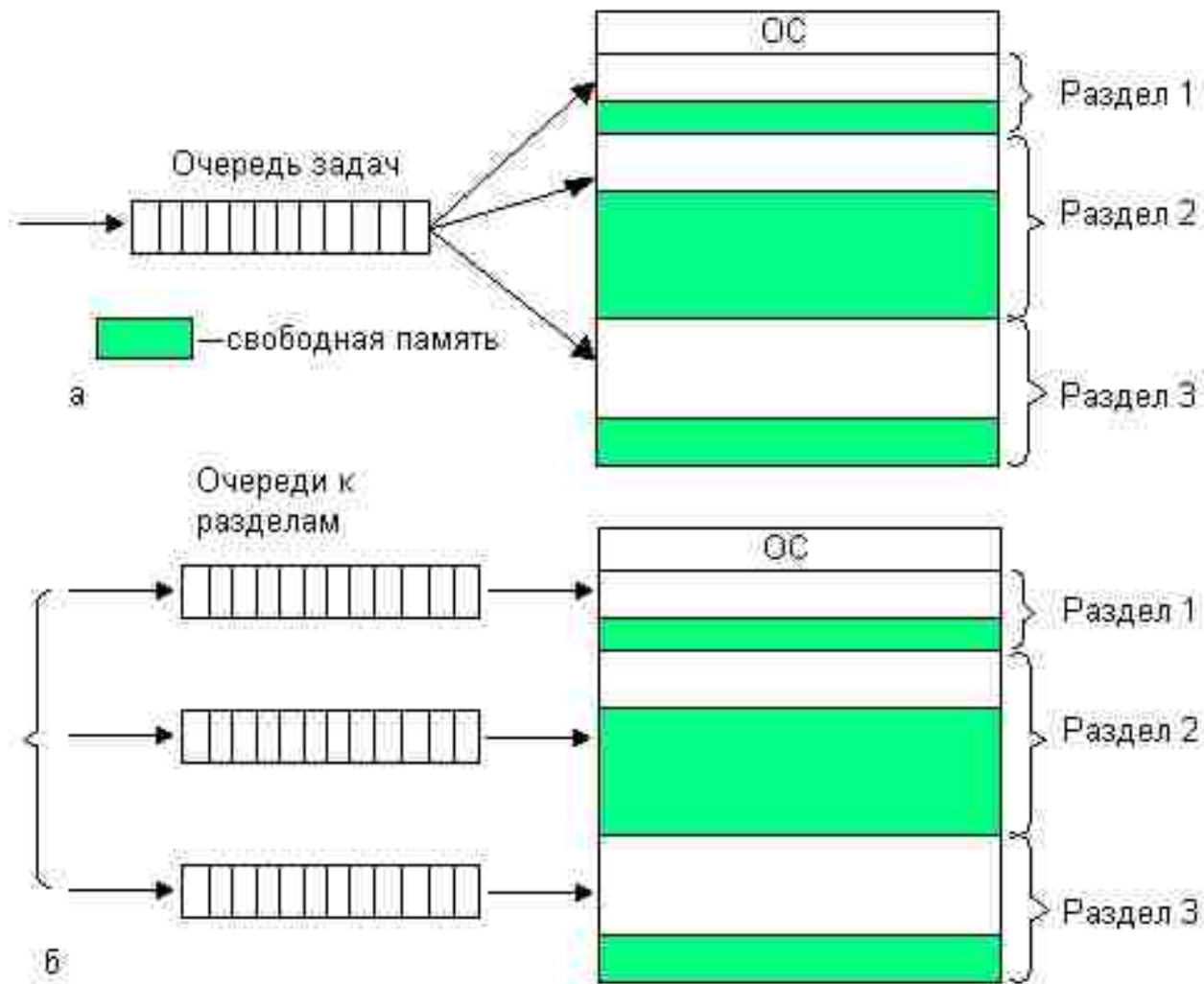



Рисунок 4 - Распределение памяти фиксированными разделами: с общей очередью (а), очередь с разделами (б)

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.

**Преимущество** - простота реализации, **существенный недостаток** - жесткость. Так как в каждом разделе может выполняться только один процесс, то уровень мультипрограммирования заранее ограничен числом разделов. Независимо от размера программы она будет занимать весь раздел.



Так, например, в системе с тремя разделами невозможно выполнять одновременно более трех процессов, даже если им требуется совсем мало памяти. С другой стороны, разбиение памяти на разделы не позволяет выполнять процессы, программы которых не помещаются ни в один из разделов, но для которых было бы достаточно памяти нескольких разделов.

Метод распределения памяти фиксированными разделами находит применение **в системах реального времени.**

## **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.

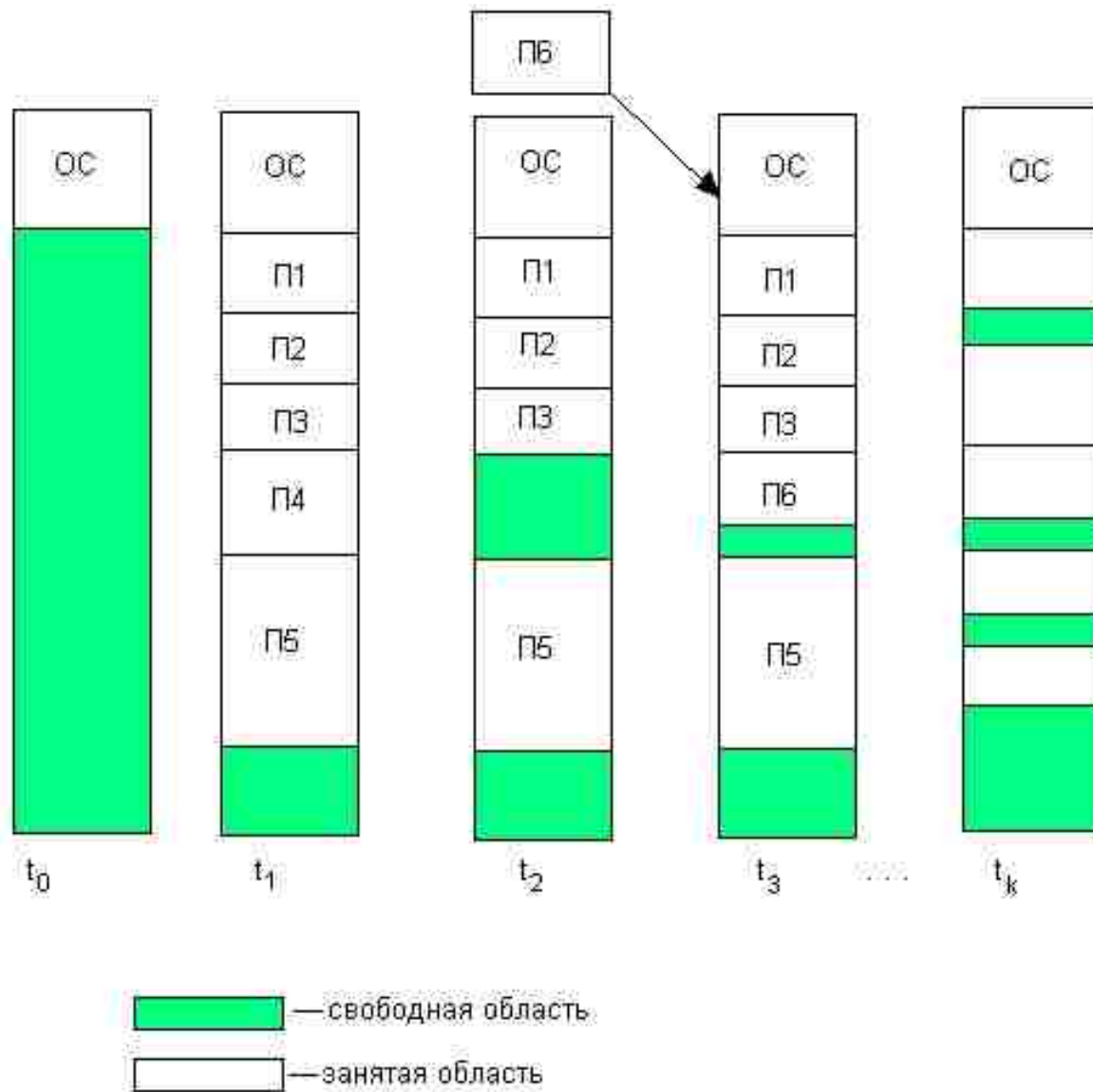



Рисунок 5 - Распределение памяти динамическими разделами



Функции операционной системы, предназначенные для реализации данного метода управления памятью:

- 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.

По сравнению с методом распределения памяти фиксированными разделами данный метод обладает гораздо большей **гибкостью**, но ему присущ очень серьезный **недостаток** - **фрагментация памяти**. **Фрагментация** - это наличие большого числа несмежных участков свободной памяти очень маленького размера (фрагментов). Настолько маленького, что ни одна из вновь поступающих программ не может поместиться ни в одном из участков, хотя суммарный объем фрагментов может составить значительную величину, намного превышающую требуемый объем памяти.

## Перемещаемые разделы.

Одним из методов борьбы с фрагментацией является перемещение всех занятых участков в сторону старших или младших адресов, так, чтобы вся свободная память образовала единую свободную область (Рис. 6). В дополнение к функциям, которые выполняет ОС при распределении памяти динамическими разделами, в данном случае она должна еще время от времени копировать содержимое разделов из одного места памяти в другое, корректируя таблицы свободных и занятых областей. Эта процедура называется **сжатием**. Сжатие может выполняться либо при каждом завершении процесса, либо только тогда, когда для вновь



создаваемого процесса нет свободного раздела  
достаточного размера.

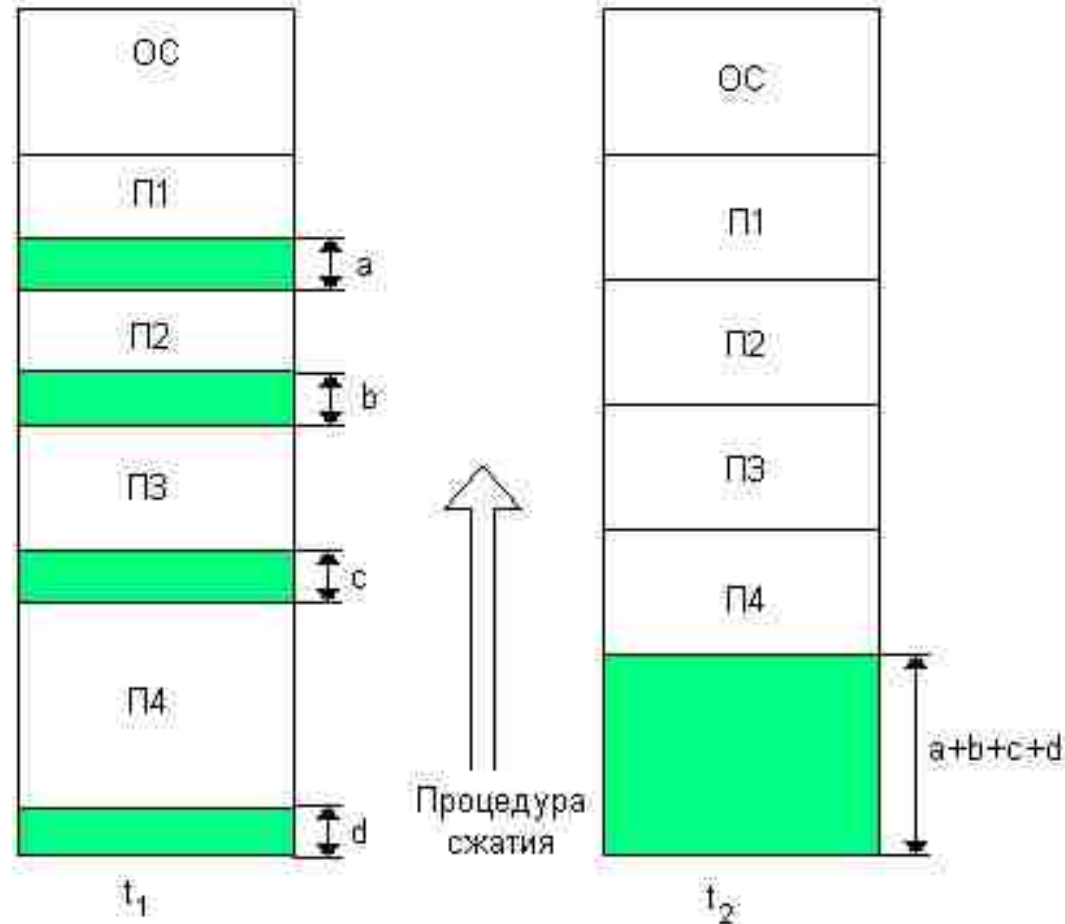



Рисунок 6 - Распределение памяти перемещаемыми разделами


## Свопинг и виртуальная память

Необходимым условием для того, чтобы программа могла выполняться, является ее **нахождение в оперативной памяти**. Только в этом случае процессор может извлекать команды из памяти и интерпретировать их, выполняя заданные действия. Объем оперативной памяти, который имеется в компьютере, существенно сказывается на характере протекания вычислительного процесса. Он ограничивает число одновременно выполняющихся программ и размеры их виртуальных адресных пространств.



Большое количество задач, необходимое для высокой загрузки процессора, требует большого объема оперативной памяти. В условиях, когда для обеспечения приемлемого уровня мультипрограммирования имеющейся оперативной памяти недостаточно, был предложен метод организации вычислительного процесса, при котором образы некоторых процессов целиком или частично временно выгружаются на диск.

**Виртуализация оперативной памяти** 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.

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.

Свопинг представляет собой частный случай виртуальной памяти и, следовательно, более

простой в реализации способ совместного использования оперативной памяти и диска.

**Недостаток** - подкачке свойственна избыточность: 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. Аналогично при освобождении памяти для загрузки нового процесса очень часто вовсе не требуется выгружать другой процесс на диск целиком, достаточно вытеснить на диск только часть его образа.

Перемещение избыточной информации замедляет работу системы, а также приводит к неэффективному использованию памяти. Кроме того, системы, поддерживающие свопинг, имеют еще один **очень существенный недостаток**: они не способны загрузить для выполнения процесс, виртуальное адресное пространство которого превышает имеющуюся в наличии свободную память. Свопинг как основной механизм управления памятью почти не используется в современных ОС. На смену ему пришел более совершенный **механизм виртуальной памяти**

В настоящее время все множество реализаций виртуальной памяти может быть представлено **тремя классами:**

- **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.

Для временного хранения сегментов и страниц на диске отводится либо специальная область, либо специальный файл, которые во многих ОС продолжают называть областью, или файлом свопинга, хотя перемещение информации между оперативной памятью и диском осуществляется уже не в форме

полного замещения одного процесса другим, а частями.

Другое популярное название этой области — **страничный файл** (page file, или paging file). Текущий размер страничного файла является важным параметром, оказывающим влияние на возможности операционной системы: чем больше страничный файл, тем больше приложений может одновременно выполнять ОС (при фиксированном размере оперативной памяти). Однако необходимо понимать, что увеличение числа одновременно работающих приложений за счет увеличения размера страничного файла замедляет их работу, так как значительная часть времени при этом тратится на перекачку кодов и данных из оперативной памяти на диск и обратно.

## Страничное распределение.

На рисунке 7 показана схема страничного распределения памяти. 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.

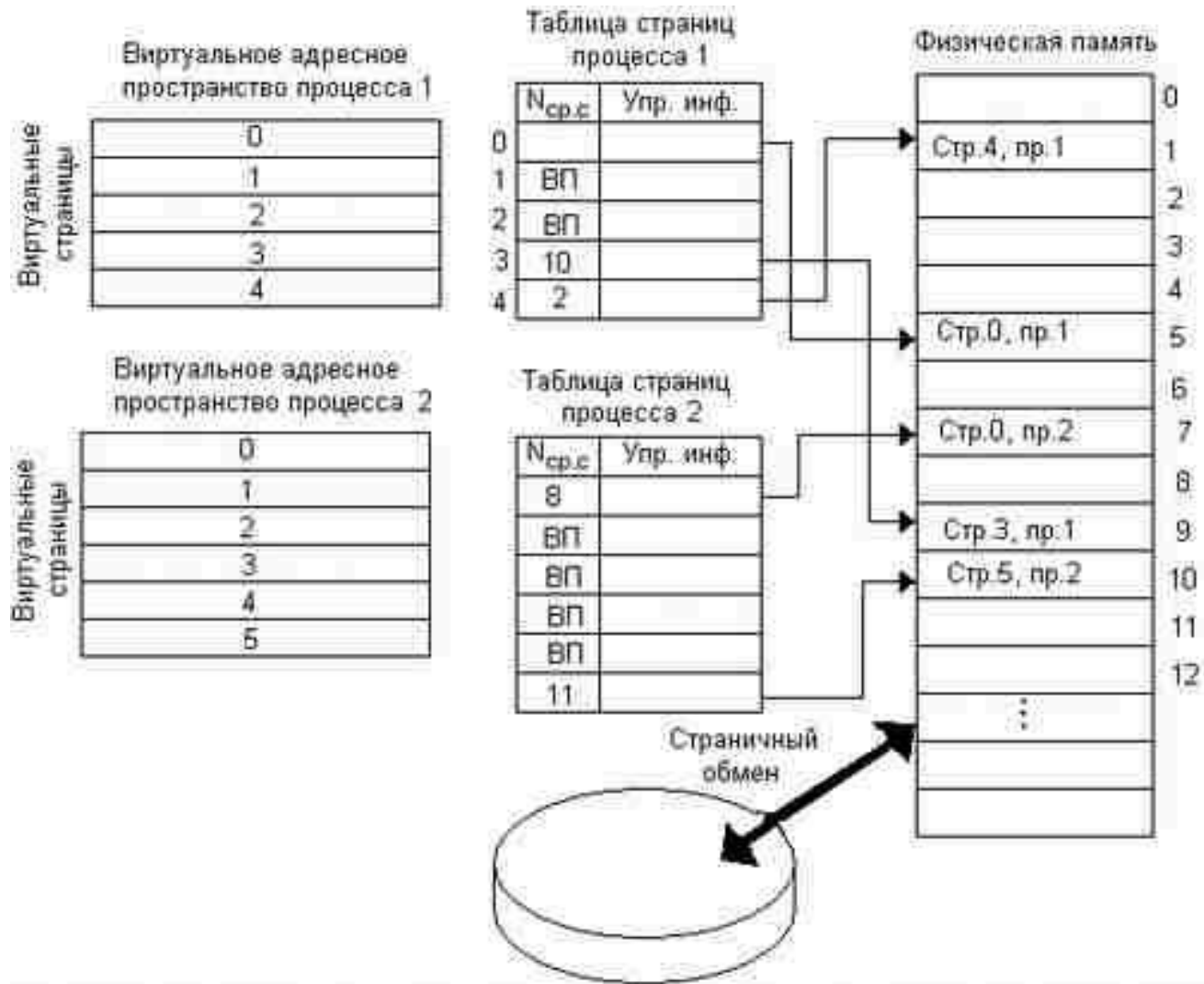



Рисунок 7 - Страничное распределение памяти



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. Это позволяет упростить механизм преобразования адресов.

При создании процесса ОС загружает в оперативную память несколько его виртуальных страниц. Копия всего виртуального адресного пространства процесса находится на диске.


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.

При каждом обращении к памяти выполняется поиск номера виртуальной страницы, содержащей требуемый адрес, затем по этому номеру определяется нужный элемент таблицы страниц, и из него извлекается описывающая страницу информация. Далее анализируется признак присутствия, и, если данная виртуальная страница находится в оперативной памяти, то выполняется преобразование виртуального адреса в физический, то есть виртуальный адрес заменяется указанным в записи таблицы физическим адресом.




Если же 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.



## Сегментное распределение.


Виртуальное адресное пространство процесса делится на части - сегменты, размер которых определяется с учетом смыслового значения содержащейся в них информации. Отдельный сегмент может представлять собой подпрограмму, массив данных и т. п. Деление виртуального адресного пространства на сегменты осуществляется компилятором на основе указаний программиста или по умолчанию, в соответствии с принятыми в системе соглашениями. Максимальный размер сегмента определяется разрядностью виртуального адреса.



На этапе создания процесса во время загрузки его образа в оперативную память система создает таблицу сегментов процесса (аналогичную таблице страниц), в которой для каждого сегмента указывается:

- базовый физический адрес сегмента в оперативной памяти;
- размер сегмента;
- правила доступа к сегменту;
- признаки модификации, присутствия и обращения к данному сегменту, а также некоторая другая информация.

**Недостатком** сегментного распределения является **избыточность**. При сегментной организации единицей перемещения между памятью и диском является сегмент, имеющий в общем случае объем больший, чем страница. Однако во многих случаях для работы программы вовсе не требуется загружать весь сегмент целиком, достаточно было бы одной или двух страниц. Аналогично при отсутствии свободного места в памяти не стоит выгружать целый сегмент, когда можно обойтись выгрузкой нескольких страниц. Но главный недостаток — это **фрагментация**, которая возникает из-за непредсказуемости размеров сегментов.



Одним из существенных отличий сегментной организации памяти от страничной является возможность задания дифференцированных прав доступа процесса к его сегментам. Например, один сегмент данных, содержащий исходную информацию для приложения, может иметь права доступа «только чтение», а сегмент данных, представляющий результаты, — «чтение и запись». Это свойство дает принципиальное **преимущество** сегментной модели памяти над страничной.

## Сегментно-страничное распределение.


Данный метод представляет собой комбинацию страничного и сегментного механизмов управления памятью и направлен на реализацию достоинств обоих подходов.

Так же как и при сегментной организации памяти, виртуальное адресное пространство процесса разделено на **сегменты**. Это позволяет определять разные права доступа к разным частям кодов и данных программы.

Перемещение данных между памятью и диском осуществляется не сегментами, а **страницами**. Для этого каждый виртуальный сегмент и физическая память делятся на страницы равного размера, что позволяет более эффективно использовать память, сократив до минимума фрагментацию.


## Кэширование данных

Память вычислительной машины представляет собой иерархию запоминающих устройств (ЗУ), отличающихся средним временем доступа к данным, объемом и стоимостью хранения одного бита (Рисунок 8). Фундаментом этой пирамиды запоминающих устройств служит внешняя память, как правило, представляемая жестким диском. Она имеет большой объем (десятки и сотни гигабайт), но скорость доступа к данным является невысокой. Время доступа к диску измеряется миллисекундами.



На следующем уровне располагается более быстродействующая (время доступа равно примерно 10-20 наносекундам) и менее объемная (от десятков мегабайт до нескольких гигабайт) оперативная память, реализуемая на относительно медленной динамической памяти DRAM.

Для хранения данных, к которым необходимо обеспечить быстрый доступ, используются компактные быстродействующие запоминающие устройства на основе статической памяти SRAM, объем которых составляет от нескольких десятков до нескольких сотен килобайт, а время доступа к данным обычно не превышает 8 нс.



Верхушку в этой пирамиде составляют внутренние регистры процессора, которые также могут быть использованы для промежуточного хранения данных. Общий объем регистров составляет несколько десятков байт, а время доступа определяется быстродействием процессора и равно в настоящее время примерно 2-3 нс.

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, с одной стороны, уменьшить среднее время доступа к данным, а с другой стороны, экономить более дорогую быстродействующую память.



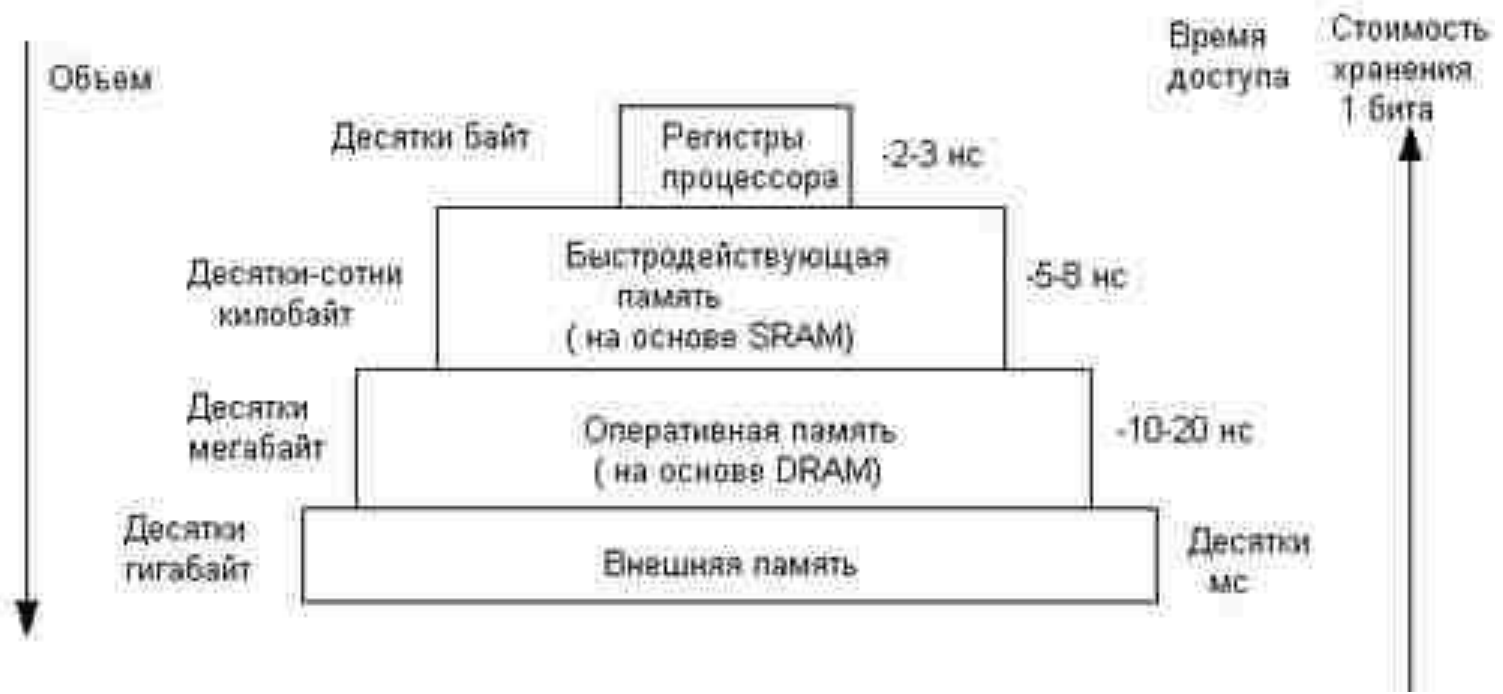


Рисунок 8 – Иерархия запоминающих устройств

## Вопросы к лекции

Каковы функции ОС по управлению памятью?

Чем отличается виртуальный адрес и физический?

Как осуществляется преобразование виртуальных адресов в физические?

На какие классы разделяются алгоритмы распределения памяти?

Как осуществляется распределение памяти фиксированными разделами?

Как осуществляется распределение памяти динамическими разделами?

Как осуществляется распределение памяти перемещаемыми разделами?

Как осуществляется страничное распределение памяти?

Как осуществляется сегментное и сегментно-страничное распределение памяти?

Что такое кэш-память и кэширование?