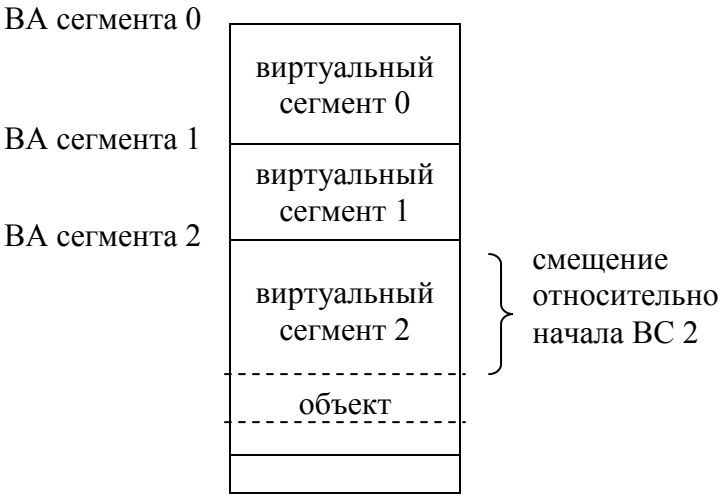
10. Сегментно-страничная организация памяти

Данный способ введен для совмещения достоинств сегментной и страничной организации. В целом, способ похож на двухуровневую страничную организацию: ВАП представляется как набор виртуальных сегментов (ВС) переменной длины, а каждый ВС разбивается на страницы фиксированной длины. Все ВС находятся в одном ВАП и нумеруются целыми числами. Каждый ВС имеет свой начальный адрес в ВАП и свой размер. Тогда ВА команды или элемента данных определяется номером ВС и смещением относительно его начала.



При создании процесса система строит **таблицу виртуальных сегментов** как набор записей-дескрипторов стандартного вида:

- виртуальный адрес начала сегмента в ВАП процесса;
- права доступа к сегменту;
- управляющая информация о сегменте, в частности – размер сегмента.

Эта таблица позволяет выполнить **первый этап** трансляции адресов: преобразовать **начальный** ВА в **промежуточный** ВА следующим образом:

- из ВА извлекается номер ВС, по которому производится вход в таблицу ВС;
- сравниваются заданное в ВА смещение и размер ВС, при необходимости генерируется прерывание;

- проверяется возможность выполнения операции в соответствии с правами доступа, установленными для сегмента, и при необходимости генерируется прерывание;
- если все нормально, то сложением ВА начала сегмента со смещением вычисляется промежуточный ВА (но пока НЕ физический адрес!).

После этого начинается **второй этап** преобразования, использующий уже **страничный** механизм. Все ВАП, как и физическая память, разбиты на страницы, обычно по 4 Кб каждая. Страницы пронумерованы сквозной нумерацией. Полученный на первом этапе промежуточный ВА разбивается на 2 части: номер страницы и смещение относительно страницы. При использовании 32−х разрядных адресов и 4−х килобайтных страниц младшие 12 разрядов промежуточного ВА определяют смещение, а старшие 20 разрядов – номер виртуальной страницы. Для каждого процесса система строит **таблицу страниц** стандартной структуры. Преобразование также выполняется стандартным образом:

- из промежуточного ВА извлекается номер виртуальной страницы, по которому производится вход в таблицу страниц;
- проверяется присутствие страницы в памяти и при необходимости генерируется страничное прерывание с выполнением всех стандартных действий;
- при необходимости проверяется доступность выполнения запрошенной операции с генерацией соответствующего прерывания;
- если все в порядке, номер виртуальной страницы заменяется номером соответствующей ей физической страницы и тем самым формируется искомый адрес.

Аналогично обычной страничной организации, недостатком данного способа является необходимость поддержки для каждого процесса большой по размеру таблицы страниц (около 4 Мбайт). Поэтому в некоторых процессорных архитектурах, и в частности – в процессорах Intel Pentium, предусматривается **двухшаговая** трансляция промежуточного ВА в ФА с

помощью промежуточных **разделов,** которая полностью соответствует описанной ранее схеме. В этом случае промежуточный ВА, полученный на первом этапе с помощью сегментного механизма, разбивается уже на **ТРИ** составляющие: номер раздела, номер страницы в разделе и смещение на странице.

В целом, вместе с сегментным механизмом здесь уже получается **трехуровневая** схема преобразования адресов: сегмент -> раздел -> страница. Конечно, это существенно усложняет архитектуру базового процессора, но дает гибкость в реализации разных механизмов управления памятью. Так, процессоры семейства Intel, реализующие именно трехуровневую сегментно-страничную организацию, позволяют при необходимости **отключать** те или иные составляющие. Можно отключить страничный механизм, получив тем самым чистую сегментную организацию. Наоборот, можно как бы выключить сегментный механизм, если реализовать код и данные процесса в **одном единственном сегменте** размером 4 Гб. Этот способ наиболее популярен в настоящее время и часто называется **“плоской (flat) моделью памяти”**. Полный трехуровневый сегментно-страничный механизм требует весьма серьезных затрат и поэтому реализуется редко, в основном – на мощных компьютерах.

Преимущества сегментно-страничной организации памяти:

- возможность использования общих (разделяемых) сегментов;
- разграничение прав доступа к сегментам;
- отсутствие фрагментации памяти за счет ее распределения небольшими страницами;
- высокая эффективность обмена с диском на страничном уровне.

Недостаток – существенно более громоздкая схема преобразования адресов, требующая хорошей аппаратной поддержки. Частично этот недостаток устраняется использованием **механизма кэширования,** когда в сверхбыстрой кэш-памяти хранятся наиболее часто используемые

дескрипторы сегментов и страниц, обращение к которым не требует обработки соответствующих таблиц.

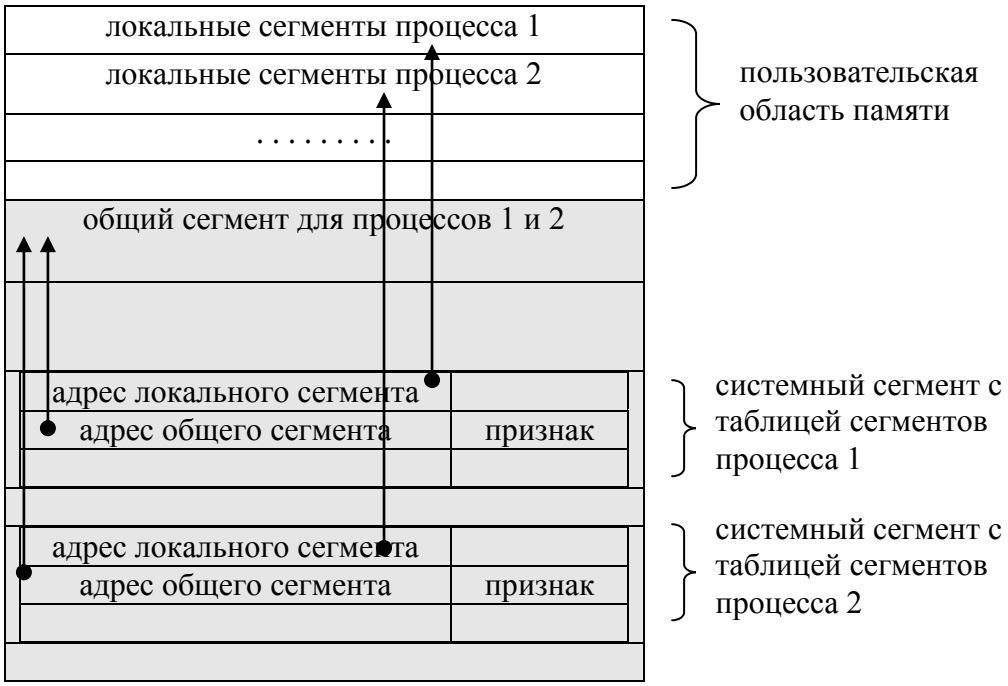
Многозадачная ОС, помимо всего, должна обеспечивать реализацию двух в некоторой степени противоречивых возможностей. С одной стороны, необходимо **защищать** код и данные процесса от воздействия со стороны других процессов. Но в то же время весьма часто **разным** процессам требуется доступ к **одним и тем же** областям памяти.

Одним из основных механизмов защиты памяти процессов является создание для каждого процесса **своих** таблиц сегментов или страниц, к которым нет доступа для кода других процессов. При переключении активного процесса происходит и переключение дескрипторных таблиц. Сами эти таблицы оформляются как системные сегменты, располагаются в системной области памяти и обрабатываются системными модулями.

Любая деятельность процесса, выходящая за рамки использования своих “родных” сегментов, должна контролироваться системой. Необходимость использования **общей** или **разделяемой** памяти (shared memory) объясняется двумя факторами:

- процессам может понадобиться обработать общую **структуру данных,** которую в этом случае надо оформить как разделяемый сегмент данных;
- код разных процессов может обращаться к **одному и тому же коду** (например, к стандартным программам), для которого также система должна создать общий сегмент.

Независимо от типа сегмента (код или данные), для доступа к ним со стороны разных процессов система должна выполнить соответствующую **настройку** дескрипторных таблиц процессов. Чаще всего, разделяемые сегменты размещаются в системной области памяти, а настройка таблиц состоит в том, что физический адрес начала сегмента или номер физической страницы в разных таблицах **одинаков.** В этом случае преобразование ВА в ФА для разных процессов дает доступ к одной и той же области памяти.



Поскольку процесс может использовать разные по типу сегменты (собственные локальные и общие), то дескрипторы сегментов или страниц должны содержать специальный флаг-признак. Общие сегменты являются **разделяемым** ресурсом, и система должна управлять ими как любым подобным ресурсом. Для этого создается специальная таблица, в которой система собирает всю необходимую информацию: какие общие сегменты предоставлены каким процессам и с какими правами доступа.

В качестве примера можно привести реализацию механизмов управления памятью в процессорах семейства Intel. Для каждого процесса создается своя **Локальная Таблица Дескрипторов** (LDT, local descriptor table), определяющая локальное адресное пространство процесса. В каждый момент времени процессор использует одну из таких таблиц, соответствующую активному процессу. На расположение активной LDT в памяти указывает специальный регистр LDTR, содержимое которого можно менять только специальными командами, доступными только в привилегированном режиме работы процессора.

Кроме множества таблиц LDT, создается **единственная** Глобальная Таблица Дескрипторов (GDT), определяющая области памяти, используемые

системой для своего кода и данных, а также – общие (разделяемые) процессами сегменты. Каждая таблица LDT хранится в своем сегменте, адрес которого содержится в таблице GDT. Расположение таблицы GDT в памяти определяется системным регистром GDTR, манипуляции с которым также разрешены только в привилегированном режиме.