9. Сегментная организация памяти

В отличие от страниц сегмент – это фрагмент адресного пространства **переменной** длины. Размер сегмента определяется транслятором исходя из его **логического назначения.** Например, сегмент с кодом главной программы, сегмент с набором подпрограмм, сегмент с обрабатываемыми данными. Это позволяет при создании программы разбить все ВАП на фрагменты-сегменты строго в соответствии с их ролью. Число одновременно поддерживаемых сегментов и их максимальный размер определяются архитектурой процессора и особенностями ОС. Так, в системе MS DOS можно было использовать одновременно до 4 сегментов длиной не более 64 Кб, тогда как в Windows размер отдельного сегмента может доходить до 4 Гб.

При создании машинного кода транслятор рассматривает сегмент как относительно самостоятельную единицу, начиная отсчет величины смещения для каждого сегмента с нуля. Другими словами, назначение адресов командам и данным внутри сегмента выполняется **независимо** от других сегментов. Программа представляет собой набор независимых сегментов разной длины. Каждому сегменту присваивается порядковый номер. Виртуальный адрес команды или элемента данных задается номером сегмента и смещением относительно начала сегмента:

ВА = (сегмент, смещение).

При запуске программы на выполнение система на основе информации, хранящейся в исполняемом файле, пытается загрузить в память все сегменты процесса. Для этого для каждого сегмента в памяти надо найти подходящее по размеру свободное место. В многозадачных системах в условиях интенсивного использования памяти многими процессами эта задача становится весьма важной и непростой, поскольку размеры всех сегментов **разные** и **заранее непредсказуемые.**

Система постоянно должна вести учет занятой и свободной памяти. Например, можно создать два динамических списка: один содержит данные о

занятых участках памяти, другой – о свободных. Каждый элемент списка должен содержать начальный адрес области памяти и ее размер. Списки могут быть упорядочены либо **по возрастанию**, либо **по убыванию** размеров областей. В первом случае поиск свободной области выполняется до тех пор, пока не будет найден **минимально подходящий** фрагмент, т.е. первый по порядку фрагмент, больший затребованного. Во втором случае **самый большой** фрагмент находится в начале списка, и если его размер больше затребованного, его можно выбрать для размещения. Этот, в какой-то степени неестественный, способ на практике показал себя лучше всех.

Если ни одного подходящего по размеру свободного фрагмента нет, то система может выполнить одно из следующих действий:

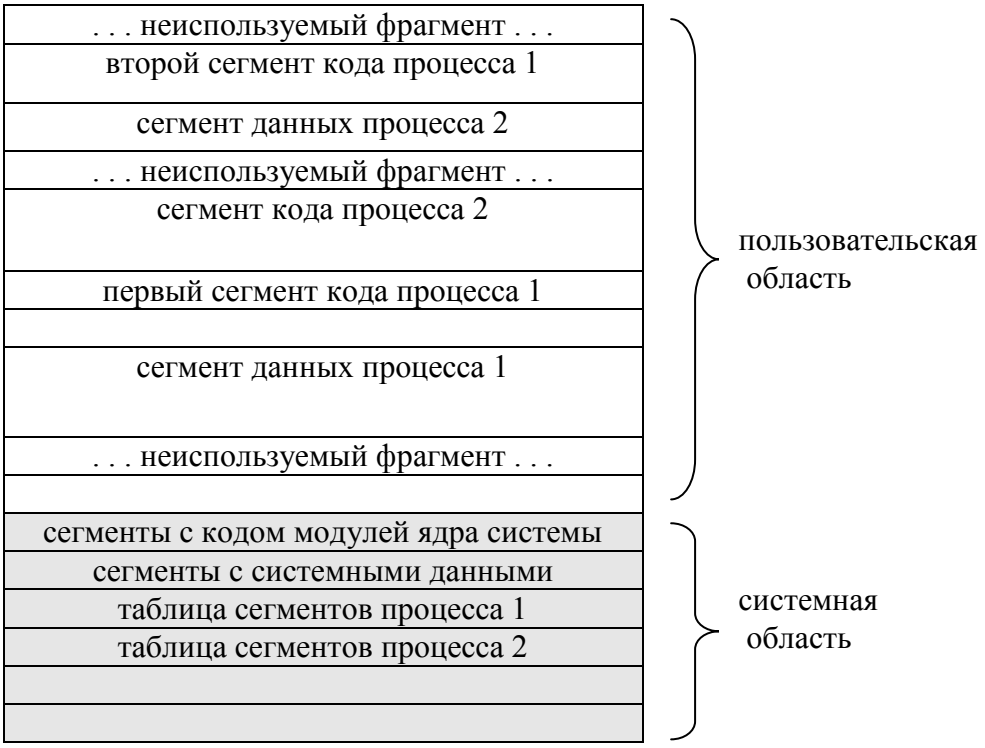
- провести **дефрагментацию** памяти, т.е. переместить все размещенные в памяти сегменты в начальные адреса строго друг за другом для устранения относительно небольших свободных “дыр” между фрагментами; эта операция достаточно трудоемкая и требует остановки всех прикладных процессов;
- **вытеснить** на диск какой-либо фрагмент, исходя из двух условий: размер фрагмента должен быть достаточным для размещения нового сегмента и фрагмент должен быть минимально используемым.

В любом случае, при запуске процесса система строит для него **таблицу сегментов** как набор записей-дескрипторов, содержащих для каждого сегмента следующие данные:

- начальный физический адрес области памяти, в которую загружается сегмент;
- размер сегмента;
- признак присутствия сегмента в памяти;
- управляющая информация, необходимая для определения вытесняемого на диск сегмента (признак модификации, частота использования);

- права доступа к сегменту (только чтение, только выполнение, чтение и запись).

Таблицы сегментов процессов сами оформляются **как сегменты,** обычно размещаемые в **системной** части основной памяти, куда доступа прикладным процессам нет. Начальный адрес сегмента, содержащего таблицу сегментов активного процесса, хранится в специальном регистре процессора. При переключении процессов это значение заменяется новым, а старое сохраняется в контексте процесса. Состояние основной памяти в некоторый момент времени схематично представлено на следующем рисунке.



Преобразование адресов при сегментной организации памяти выполняется следующим образом:

- из ВА извлекается номер сегмента, по которому производится вход в таблицу сегментов;
- в выбранной записи-дескрипторе проверяется признак присутствия сегмента в памяти;

- если сегмента в памяти нет, генерируется прерывание, обработчик которого выполняет поиск и загрузку затребованного сегмента с вытеснением при необходимости “лишнего” сегмента на диск;
- проверяется возможность выполнения текущей команды в соответствии с правами доступа, установленными для сегмента; если выполнение команды запрещено, генерируется специальное прерывание;
- из ВА извлекается смещение и сравнивается с длиной сегмента: если смещение больше длины сегмента, также генерируется прерывание ;
- если все нормально, то из дескриптора сегмента извлекается начальный адрес сегмента, который складывается со смещением, и тем самым определяется физический адрес команды или элемента данных.

Достоинствами сегментной организации являются:

- логическое разбиение кода и данных на фрагменты-сегменты;
- простота реализации доступа разных процессов к одному и тому же общему (разделяемому) сегменту;
- относительно небольшие накладные расходы на поддержку таблиц сегментов (число сегментов значительно меньше числа страниц).

Недостатки сегментной организации:

- более сложное преобразование ВА в ФА, требующее выполнения операции сложения;
- высокая фрагментация памяти;
- обмен с диском на уровне достаточно больших сегментов.