ГОСУДАРСТВЕННЫЙ КОМИТЕТ СВЯЗИ, ИНФОРМАТИЗАЦИИ И ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИИ
ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КУРСОВАЯ РАБОТА

ТЕМА: Сетевое распределение

Выполнил (а): Хожаниязов Ю. ГЧ2-22 у.

Проверил (а) Джабаева А.

Ташкент 2014

Сегментное распределение

При страничной организации виртуальное адресное пространство процесса делится на равные части механически, без учета смыслового значения данных. В одной странице могут оказаться и коды команд, и инициализируемые переменные, и массив исходных данных программы. Такой подход не позволяет обеспечить дифференцированный доступ к разным частям программы, а это свойство могло бы быть очень полезным во многих случаях. Например, можно было бы запретить обращаться с операциями записи в сегмент программы, содержащий коды команд, разрешив эту операцию для сегментов данных.

Кроме того, разбиение виртуального адресного пространства на «осмысленные» части делает принципиально возможным совместное использование фрагментов программ разными процессами. Пусть, например, двум процессам требуется одна и та же подпрограмма, которая к тому же обладает свойством реентерабельности . Тогда коды этой подпрограммы могут быть оформлены в виде отдельного сегмента и включены в виртуальные адресные пространства обоих процессов. При отображении в физическую память сегменты, содержащие коды подпрограммы из обоих виртуальных пространств, проецируются на одну и ту же область физической памяти. Таким

образом оба процесса получат доступ к одной и той же копии подпрограммы (рис. 5.18).

Итак, 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, например при 32-разрядной организации процессора он равен 4 Гбайт. При этом максимально возможное виртуальное адресное пространство процесса представляет собой набор из N виртуальных сегментов, каждый размером по 4 Гбайт. В каждом сегменте виртуальные адреса находятся в диапазоне от 0000000016 до FFFFFFFF16. Сегменты не упорядочиваются друг относительно друга, так что общего для сегментов линейного виртуального адреса не существует, виртуальный адрес задается парой чисел: номером сегмента и линейным виртуальным адресом внутри сегмента.

1 Реентерабельность (reentrantable) — свойство повторной входимости кода, которое позволяет одновременно использовать его несколькими

процессами. При выполнении реентерабельного кода процессы не изменяют его, поэтому в память достаточно загрузить только одну копию кода.

При загрузке процесса в оперативную память помещается только часть его сегментов, полная копия виртуального адресного пространства находится в дисковой памяти. Для каждого загружаемого сегмента операционная система подыскивает непрерывный участок свободной памяти достаточного размера. Смежные в виртуальной памяти сегменты одного процесса могут занимать в оперативной памяти несмежные участки. Если во время выполнения процесса происходит обращение по виртуальному адресу, относящемуся к сегменту, который в данный момент отсутствует в памяти, то происходит прерывание. ОС приостанавливает активный процесс, запускает на выполнение следующий процесс из очереди, а параллельно организует загрузку нужного сегмента с диска. При отсутствии в памяти места, необходимого для загрузки сегмента, операционная система выбирает сегмент на выгрузку, при этом она использует критерии, аналогичные рассмотренным выше критериям выбора страниц при страничном способе управления памятью.

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.

Если виртуальные адресные пространства нескольких процессов включают один и тот же сегмент, то в таблицах сегментов этих процессов делаются ссылки на один и тот же участок оперативной памяти, в который данный сегмент загружается в единственном экземпляре.

Как видно, сегментное распределение памяти имеет очень много общего со страничным распределением.

Механизмы преобразования адресов этих двух способов управления памятью тоже весьма схожи, однако в них имеются и существенные отличия, которые являются следствием того, что сегменты в отличие от страниц имеют произвольный размер. Виртуальный адрес при сегментной организации памяти может быть представлен парой (g, s), где g — номер сегмента, а s — смещение в сегменте. Физический адрес получается путем сложения базового адреса сегмента, который

определяется по номеру сегмента g из таблицы сегментов и смещения

В данном случае нельзя обойтись операцией конкатенации, как это

делается при страничной организации памяти. Действительно,

поскольку размер страницы равен степени двойки, следовательно, в

двоичном виде он выражается числом с несколькими нулями в

младших разрядах. Страницы имеют одинаковый размер, а значит, их

начальные адреса кратны размеру страниц и выражаются также

числами с нулями в младших разрядах. Именно поэтому ОС заносит в

таблицы страниц не полные адреса, а номера физических страниц,

которые совпадают со старшими разрядами базовых адресов. Сегмент

же может в общем случае располагаться в физической памяти начиная

с любого адреса, следовательно, для определения местоположения в

памяти необходимо задавать его полный начальный физический

адрес. Использование операции сложения вместо конкатенации

замедляет процедуру преобразования виртуального адреса в

физический по сравнению со страничной организацией.

Другим 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 сегментного распределения — это фрагментация, которая возникает из-за непредсказуемости размеров сегментов. В процессе работы системы в памяти образуются небольшие участки свободной памяти, в которые не может быть загружен ни один сегмент. Суммарный объем, занимаемый фрагментами, может составить существенную часть общей памяти системы, приводя к ее неэффективному использованию.

Система с сегментной организацией функционирует аналогично системе со страничной организацией: при каждом обращении к оперативной памяти выполняется преобразование виртуального адреса в физический, время от времени происходят прерывания, связанные с отсутствием нужных сегментов в памяти, при необходимости освобождения памяти некоторые сегменты выгружаются.

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.

В большинстве современных реализаций сегментно-страничной организации памяти в отличие от набора виртуальных диапазонов адресов при сегментной организации памяти (рис. 5.20, а) все виртуальные сегменты образуют одно непрерывное линейное виртуальное адресное пространство (рис. 5.20, б).

Координаты байта в виртуальном адресном пространстве при сегментно-страничной организации можно задать двумя способами. Во-первых, линейным виртуальным адресом, который равен сдвигу данного байта относительно границы общего линейного виртуального пространства, во-вторых, парой чисел, одно из которых является номером сегмента, а другое — смещением относительно начала сегмента. При этом в отличие от сегментной модели, для однозначного задания виртуального адреса вторым способом необходимо каким-то образом указать также начальный виртуальный адрес сегмента с данным номером. Системы виртуальной памяти ОС с сегментно-страничной организацией используют второй способ, так как он позволяет непосредственно определить принадлежность адреса некоторому сегменту и проверить права доступа процесса к нему.

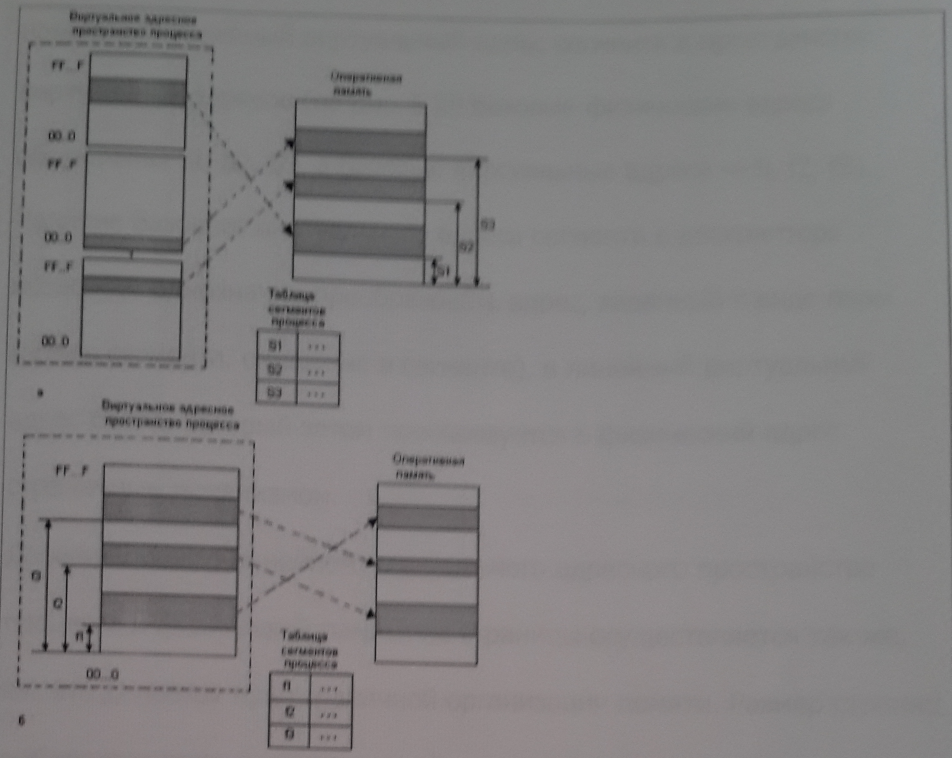

Рис. 5.20. Два способа сегментации

Для каждого процесса операционная система создает отдельную таблицу сегментов, в которой содержатся описатели (дескрипторы) всех сегментов процесса. Описание сегмента включает назначенные ему права доступа и другие характеристики, подобные тем, которые содержатся в дескрипторах сегментов при сегментной организации памяти. Однако имеется и принципиальное отличие. В поле базового адреса указывается не начальный физический адрес сегмента, отведенный ему в результате загрузки в оперативную память, а

начальный линейный виртуальный адрес сегмента в пространстве виртуальных адресов (на рис. 5.20 базовые физические адреса обозначены SI, S2, S3, а базовые виртуальные адреса — fl, f2, f3). Наличие базового виртуального адреса сегмента в дескрипторе позволяет однозначно преобразовать адрес, заданный в виде пары (номер сегмента, смещение в сегменте), в линейный виртуальный адрес байта, который затем преобразуется в физический адрес страничным механизмом.

Деление общего линейного виртуального адресного пространства процесса и физической памяти на страницы осуществляется так же, как это делается при страничной организации памяти. Размер страниц выбирается равным степени двойки, что упрощает механизм преобразования виртуальных адресов в физические. Виртуальные страницы нумеруются в пределах виртуального адресного пространства каждого процесса, а физические страницы — в пределах оперативной памяти. При создании процесса в память загружается только часть страниц, остальные загружаются по мере необходимости. Время от времени система выгружает уже ненужные страницы, освобождая память для новых страниц. ОС ведет для каждого процесса таблицу страниц, в которой указывается соответствие виртуальных страниц физическим.

Базовые адреса таблицы сегментов и таблицы страниц процесса являются частью его контекста. При активизации процесса эти адреса загружаются в специальные регистры процессора и используются механизмом преобразования адресов.

Преобразование виртуального адреса в физический происходит в два этапа (рис. 5.21):

1. На первом этапе работает механизм сегментации. Исходный виртуальный адрес, заданный в виде пары (номер сегмента, смещение), преобразуется в линейный виртуальный адрес. Для этого на основании базового адреса таблицы сегментов и номера сегмента вычисляется адрес дескриптора сегмента. Анализируются поля дескриптора и выполняется проверка возможности выполнения заданной операции. Если доступ к сегменту разрешен, то вычисляется линейный виртуальный адрес путем сложения базового адреса сегмента, извлеченного из дескриптора, и смещения, заданного в исходном виртуальном адресе.

2. На втором этапе работает страничный механизм. Полученный линейный виртуальный адрес преобразуется в искомый физический адрес. В результате преобразования линейный виртуальный адрес представляется в том виде, в котором он используется при страничной организации памяти, а именно в виде пары (номер страницы,

смещение в странице). Благодаря тому что размер страницы выбран равным степени двойки, эта задача решается простым отделением некоторого количества младших двоичных разрядов. При этом в старших разрядах содержится номер виртуальной страницы, а в младших — смещение искомого элемента относительно начала страницы. Так, если размер страницы равен 2k, то смещением является содержимое младших k разрядов, а остальные, старшие разряды содержат номер виртуальной страницы, которой принадлежит искомый адрес. Далее преобразование адреса происходит так же, как при страничной организации: старшие разряды линейного виртуального адреса, содержащие номер виртуальной страницы, заменяются номером физической страницы, взятым из таблицы страниц, а младшие разряды виртуального адреса, содержащие смещение, остаются без изменения.

Как видно, механизм сегментации и страничный механизм действуют достаточно независимо друг от друга. Поэтому нетрудно представить себе реализацию сегментно-страничного управления памятью, в которой механизм сегментации работает по вышеописанной схеме, а страничный механизм изменен. Он реализует двухуровневую схему, в которой виртуальное адресное пространство делится сначала на разделы, а уж потом на страницы. В таком случае преобразование

виртуального адреса в физический происходит в несколько этапов. Сначала механизм сегментации обычным образом, используя таблицу сегментов, вычисляет линейный виртуальный адрес. Затем из данного виртуального адреса вычленяются номер раздела, номер страницы и смещение. И далее по номеру раздела из таблицы разделов определяется адрес таблицы страниц, а затем по номеру виртуальной страницы из таблицы страниц определяется номер физической страницы, к которому пристыковывается смещение.Именно такой подход реализован компанией Intel в процессорах 1386, i486 и Pentium.

Рассмотрим еще одну возможную схему управления памятью, основанную на комбинировании сегментного и страничного механизмов. Так же как и в предыдущих случаях, виртуальное пространство процесса делится на сегменты, а каждый сегмент, в свою очередь, делится на виртуальные страницы. Первое отличие состоит в том, что виртуальные страницы нумеруются не в пределах всего адресного пространства процесса, а в пределах сегмента. Виртуальный адрес в этом случае выражается тройкой (номер сегмента, номер страницы, смещение в странице).

Загрузка процесса выполняется операционной системой постранично, при этом часть страниц размещается в оперативной памяти, а часть —

на диске. Для каждого процесса создается собственная таблица сегментов, а для каждого сегмента — своя таблица страниц. Адрес таблицы сегментов загружается в специальный регистр процессора, когда активизируется соответствующий процесс.

Таблица страниц содержит дескрипторы страниц, содержимое которых полностью аналогично содержимому ранее описанных дескрипторов страниц. А вот таблица сегментов состоит из дескрипторов сегментов, которые вместо информации о расположении сегментов в виртуальном адресном пространстве содержат описание расположения таблиц страниц в физической памяти. Это является вторым существенным отличием данного подхода от ранее рассмотренной схемы сегментно-страничной организации.

1. По номеру сегмента, заданному в виртуальном адресе, из таблицы сегментов извлекается физический адрес соответствующей таблицы страниц.

2. По номеру виртуальной страницы, заданному в виртуальном адресе, из таблицы страниц извлекается дескриптор, в котором указан номер физической страницы.

3. К номеру физической страницы пристыковывается младшая часть виртуального адреса — смещение.

Разделяемые сегменты памяти

Подсистема виртуальной памяти представляет собой удобный механизм для решения задачи совместного доступа нескольких процессов к одному и тому же сегменту памяти, который в этом случае называется разделяемой памятью (shared memory).

Хотя основной задачей операционной системы при управлении памятью является защита областей оперативной памяти, принадлежащей одному из процессов, от доступа к ней остальных процессов, в некоторых случаях оказывается полезным организовать контролируемый совместный доступ нескольких процессов к определенной области памяти. Например, в том случае, когда несколько пользователей одновременно работают с некоторым текстовым редактором, нецелесообразно многократно загружать его код в оперативную память. Гораздо экономичней загрузить всего одну копию кода, которая обслуживала бы всех пользователей, работающих в данное время с этим редактором (для этого код редактора должен быть реентерабельным). Очевидно, что сегмент данных редактора не может присутствовать в памяти в единственном разделяемом экземпляре — для каждого пользователя должна быть создана своя копия этого сегмента, в которой помещается редактируемый текст и значения других переменных редактора, например его конфигурация,

индивидуальная для каждого пользователя, и т. п.

Другим примером применения разделяемой области памяти может быть использование ее в качестве буфера при межпроцессном обмене данными. В этом случае один процесс пишет в разделяемую область, а другой — читает.

Для организации разделяемого сегмента при наличии подсистемы виртуальной памяти достаточно поместить его в виртуальное адресное пространство каждого процесса, которому нужен доступ к данному сегменту, а затем настроить параметры отображения этих виртуальных сегментов так, чтобы они соответствовали одной и той же области оперативной памяти. Детали такой настройки зависят от типа используемой в ОС модели виртуальной памяти: сегментной или сегментно-страничной (чисто страничная организация не поддерживает понятие «сегмент», что делает невозможным решение рассматриваемой задачи). Например, при сегментной организации необходимо в дескрипторах виртуального сегмента каждого процесса указать один и тот же базовый физический адрес. При сегментно-страничной организации отображение на одну и ту же область памяти достигается за счет соответствующей настройки таблицы страниц каждого процесса.

В приведенном выше описании подразумевалось, что разделяемый

сегмент помещается в индивидуальную часть виртуального адресного пространства каждого процесса (рис. 5.23, а) и описывается в каждом процессе индивидуальным дескриптором сегмента (и индивидуальными дескрипторами страниц, если используется сегментно-страничный механизм). «Попадание» же этих виртуальных сегментов на общую часть оперативной памяти достигается за счет согласованной настройки операционной системой многочисленных дескрипторов для множества процессов.

Возможно и более экономичное для ОС решение этой задачи — помещение единственного разделяемого виртуального сегмента в общую часть виртуального адресного пространства процессов, то есть в ту часть, которая обычно используется для модулей ОС (рис. 5.23, б). В этом случае настройка дескриптора сегмента (и дескрипторов страниц) выполняется только один раз, а все процессы пользуются такой настройкой и совместно используют часть оперативной памяти.

При работе с разделяемыми сегментами памяти ОС должна выполнять некоторые функции, общие для любых разделяемых между процессами ресурсов — файлов, семафоров и т. п. Эти функции состоят в поддержке схемы именования ресурсов, проверке прав доступа определенного процесса к ресурсу, а также в отслеживании количества процессов, пользующихся данным ресурсом (чтобы

удалить его в случае ненадобности). Для того чтобы отличать разделяемые сегменты памяти от индивидуальных, дескриптор сегмента должен содержать поле, имеющее два значения: shared (разделяемый) или private (индивидуальный).

Операционная система может создавать разделяемые сегменты как по явному запросу, так и по умолчанию. В первом случае прикладной процесс должен выполнить соответствующий системный вызов, по которому операционная система создает новый сегмент в соответствии с указанными в вызове параметрами: размером сегмента, разрешенными над ним операциями (чтение/запись) и идентификатором. Все процессы, выполнившие подобные вызовы с одним и тем же идентификатором, получают доступ к этому сегменту и используют его по своему усмотрению, например в качестве буфера для обмена данными.

Во втором случае операционная система сама в определенных ситуациях принимает решение о том, что нужно создать разделяемый сегмент. Наиболее типичным примером такого рода является поступление нескольких запросов на выполнение одного и того же приложения. Если кодовый сегмент приложения помечен в исполняемом файле как реентерабельный и разделяемый, то ОС не создает при поступлении нового запроса новую индивидуальную для

процесса копию кодового сегмента этого приложения, а отображает уже существующий разделяемый сегмент в виртуальное адресное пространство процесса. При закрытии приложения каким-либо процессом ОС проверяет, существуют ли другие процессы, пользующиеся данным приложением, и если их нет, то удаляет данный разделяемый сегмент.

Разделяемые сегменты выгружаются на диск системой виртуальной памяти по тем же алгоритмам и с помощью тех же механизмов, что и индивидуальные.

Кэширование данных

Иерархия запоминающих устройств

Память вычислительной машины представляет собой иерархию запоминающих устройств (ЗУ), отличающихся средним временем доступа к данным, объемом и стоимостью хранения одного бита (рис. 5.24). 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 доступа1 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.

1 Все перечисленные характеристики ЗУ быстро изменяются по мере совершенствования вычислительной аппаратуры. В данном случае важны не абсолютные значения времени доступа или объема памяти, а их соотношение для разных типов Запоминающих устройств.

И наконец, 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.

Таким образом, можно констатировать печальную закономерность — чем больше объем устройства, тем менее быстродействующим оно является. Более того, стоимость хранения данных в расчете на один

бит также увеличивается с ростом быстродействия устройств. Однако пользователю хотелось бы иметь и недорогую, и быструю память. Кэш-память представляет некоторое компромиссное решение этой проблемы.