Лекция 17. Виртуальная память

17.1 Понятие виртуальной памяти

Для большинства типичных применений ВМ характерна ситуация, когда размещение всей программы в ОП невозможно из-за ее большого размера. В этом, однако и нет принципиальной необходимости, поскольку в каждый момент времени «внимание» машины концентрируется на определенных сравнительно небольших участках программы. Таким образом, в ОП достаточно хранить только используемые данный период части программ, а остальные части могут располагаться на внешних ЗУ (ВЗУ). Сложность подобного подхода в том, что процессы обращения к ОП и ВЗУ существенно различаются, и это усложняет задачу программиста. Выходом из такой ситуации было появление в 1959 году идеи *виртуализации памяти* под которой понимается метод автоматического управления иерархической памятью, при котором программисту кажется, что он имеет дело с единой памятью большой емкости и высокого быстродействия. Эту память называют виртуальной (кажущейся) памятью. По своей сути виртуализация памяти представляет собой способ аппаратной и программной реализации концепции иерархической памяти.

В рамках идеи виртуализации памяти ОП рассматривается как линейное пространство *N* адресов, называемое *физическим пространством* памяти. Для задач где требуется более чем *N* ячеек, предоставляется значительно большее пространство адресов (обычно равное общей емкости всех видов памяти), называемое *виртуальным пространством,* в общем случае не обязательно линейное. Адреса виртуального пространства называют *виртуальными,* а адреса физического пространства — *физическими.* Программа пишется в виртуальных адресах, но поскольку для ее выполнения нужно, чтобы обрабатываемые команды и данные находились в ОП, требуется, чтобы каждому виртуальному адресу соответствовал физический. Таким образом, в процессе вычислений необходимо, прежде всего, переписать из ВЗУ в ОП ту часть информации, на которую указывает виртуальный адрес (отобразить виртуальное пространство на физическое), после чего преобразовать виртуальный адрес в физический (рис. 17.1.1).

Рис. 17.1.1 Отображение виртуального адреса на физический

Среди систем виртуальной памяти можно выделить два класса: системы с фиксированным размером блоков (страничная организация) и системы с переменным размером блоков (сегментная организация). Оба варианта обычно совмещают (сегментно-страничная организация).

17.2 Страничная организация памяти

Целям преобразования виртуальных адресов в физические служит страничная организация памяти. Ее идея состоит в разбиении программы на части равной величины, называемые *страницами.* Размер страницы обычно выбирают в предела-4-8 Кбайт, но так, чтобы он был кратен емкости одного сектора магнитного диска. Виртуальное и физическое адресные пространства разбиваются на блоки размером в страницу. Блок основной памяти, соответствующий странице, часто называю *страничным кадром* или *фреймом* (page frame). Страницам виртуальной и физической памяти присваивают номера. Процесс доступа к данным по их виртуальному адресу иллюстрирует рис. 17.2.1.

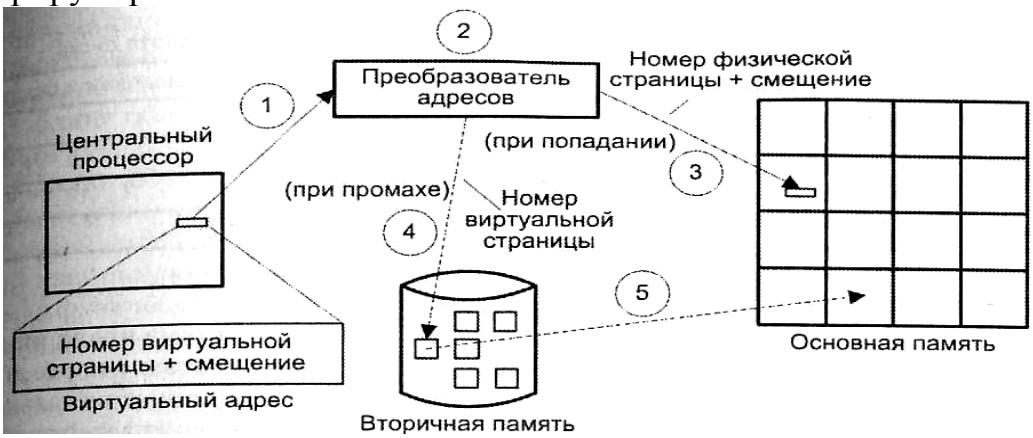

Рис. 17.2.1 Страничная организация виртуальной памяти

Центральный процессор обращается к ячейке, указав ее виртуальный адрес ①, состоящий из номера виртуальной страницы и смещения относительно ее начала. Этот адрес поступает в систему преобразования адресов②, с целью получения из него физического адреса ячейки в основной памяти ③.. Поскольку смещение в виртуальном и физическом адресе одинаковое, преобразованию подвергается лишь номер страницы. Если преобразователь обнаруживает, что нужная физическая страница отсутствует в основной памяти (произошел промах или страничный сбой), то нужная страница считывается из внешней памяти и заносится в ОП (④, ⑤) .

Преобразователь адресов — это часть операционной системы, транслирующая номер виртуальной страницы в номер физической страницы, расположенной в основной памяти, а также аппаратура, обеспечивающая этот процесс и позволяющая ускорить его. Преобразование осуществляется с помощью так называемой *страничной таблицы.* При отсутствии нужной страницы в ОП преобразователь адресов вырабатывает признак страничного сбоя, по которому операционная система приостанавливает вычисления, пока нужная страница не будет считана из вторичной памяти и помещена в основную.

Виртуальное пространство полностью описывается двумя таблицами: страничной таблицей и картой диска (будем считать, что вторичная память реализована на магнитных дисках). Таблица страниц определяет, какие виртуальные страницы находятся в основной памяти и в каких физических

фреймах, а карта диска содержит информацию о секторах диска, где хранятся виртуальные страницы на диске.

Число записей в страничной таблице (СТ) в общем случае равно количеству виртуальных страниц. Каждая запись содержит поле номера физической страницы (НФС) и четыре признака: V, R, M и A.

Признак присутствия V устанавливается в единицу, если виртуальная страница в данный момент находится в основной памяти. В этом случае в поле номера физической страницы находится соответствующий номер. Если V = 0, то при попытке обратиться к данной виртуальной странице преобразователь адреса генерирует сигнал страничного сбоя (page fault), и операционная система предпринимает действия по загрузке страницы с диска в ОП, обращаясь для этого к карте диска В карте указано, на какой дорожке и в каком секторе диска расположена каждая из виртуальных страниц[1]. Загрузка страницы с диска в ОП сопровождается записью в соответствующую строку страничной таблицы (указывается номер физической страницы, куда была загружена виртуальная страница).

В принципе карта диска может быть совмещена со страничной таблицей путем добавления к последней еще одного поля. Другим вариантом может быть увеличение разрядности поля номера физической страницы и хранение в нем номеров дорожек и секторов для виртуальных страниц, отсутствующих в основной памяти. В этом случае вид хранимой информации будет определять признак V.

Признак использования страницы R устанавливается при обращении к данной станице. Эта информация используется в алгоритме замещения страниц для выбора той из них, которую можно наиболее безболезненно удалить из ОП, чтобы освободить место для новой. Проблемы замещения информации в ОП решаются так же, как и для кэш-памяти.

Поскольку в ОП находятся лишь копии страниц, а их оригиналы хранятся на диске, необходимо обеспечить идентичность подлинников и копий. В ходе вычислений содержимое отдельных страниц может изменяться, что фиксируется путем установки в единицу *признака модификации* M. При удалении страницы из ОП проверяется состояние признака M. Если M = 1, то перед удалением страницы из основной памяти ее необходимо переписать на диск, а при M = 0 этого можно не делать.

Признак прав доступа A служит целям защиты информации и определяет, какой вид доступа к странице разрешен: только для чтения, только для записи, для чтения и для записи.

Когда программа загружается в ОП, она может быть направлена в любые свободные в данный момент страничные кадры, независимо от того, расположены они подряд или нет. Страничная организация позволяет сократить объем пересылок информации между внешней памятью и ОП, так как страницу не нужно загружать до тех пор, пока она действительно не понадобится.

Способ реализации СТ жизненно важен для эффективности техники виртуальной адресации, поскольку каждое обращение к памяти предполагает

обращение к страничной таблице. Наиболее быстрый способ — хранение таблицы в специально выделенных для этого регистрах, но от него приходится отказываться при большом объеме СТ. Остается практически один вариант — выделение страничной таблице области основной памяти, несмотря на то что это приводит к двукратному увеличению времени доступа к информации. Чтобы сократить это время в состав ВМ включают дополнительное ЗУ, называемое *буфером быстрого преобразования адреса* (TLB — Translation Look-aside Buffer), или *буфером ассоциативной трансляции,* или *буфером опережающей выборки* и представляющее собой кэш-память. При каждом преобразовании номера виртуальной страницы в номер физической страницы результат заносится в TLB: номер физической страницы в память данных, а виртуальной — в память тегов. Таким образом, в TLB попадают результаты нескольких последних операций трансляции адресов. При каждом обращении к ОП преобразователь адресов сначала производит поиск в памяти тегов TLB номера требуемой виртуальной страницы. При попадании адрес соответствующей физической страницы берется из памяти данных TLB. Если в TLB зафиксирован промах, то процедура преобразования адресов производится с помощью страничной таблицы, после чего осуществляется запись новой пары «номер виртуальной страницы — номер физический страницы» в TLB.

Буфер преобразования адресов обычно реализуется в виде полностью ассоциативной или множественно-ассоциативной кэш-памяти с высокой степенью ассоциативности и временем доступа, сопоставимым с аналогичным показателем для кэш-памяти первого уровня (L1).

Серьезной проблемой в системе виртуальной памяти является большой объем страничных таблиц, который пропорционален числу виртуальных страниц. Таблица занимает значительную часть ОП, а на поиск уходит много времени, что крайне нежелательно.

Один из способов сокращения длины таблиц основан на введении многоуровневой организации таблиц. В этом варианте информация оформляется в виде нескольких страничных таблиц сравнительно небольшого объема, которые образуют второй уровень. Первый уровень представлен таблицей с каталогом, где указано месторасположение каждой из страничных таблиц (адрес начала таблицы в памяти) второго уровня. Сначала в каталоге определяется расположение нужной страничной таблицы и лишь затем производится обращение к нужной таблице. О достигаемом эффекте можно судить из следующего примера. Пусть адресная шина ВМ имеет ширину 32 бита, а размер страницы равен 4 Кбайт. Тогда количество виртуальных страниц, а следовательно, и число входов в единой страничной таблице составит 2^{20}. При двухуровневой организации можно обойтись одной страницей первого уровня на 1024 (2^{10}) входов и 1024 страничными таблицами на такое же число входов.

Другой подход называют *способом обращенных* или *инвертированных страничных таблиц.* Такую таблицу в каком-то смысле можно рассматривать как увеличенный эквивалент TLB, отличающийся тем, что она хранится в ОП

и реализуется не аппаратурой, а программными средствами. Число входов в таблицу определяется емкостью ОП и равно числу страниц, которое может быть размещено в основной памяти. Одновременно с этим имеется и традиционная СТ, но хранится она не в ОП, а на диске. Для поиска нужной записи в инвертированной таблице используется хэширование, когда номер записи в таблице вычисляется в соответствии с некой хэш-функцией. Аргументом этой функции служит номер искомой виртуальной страницы. Хэширование позволяет ускорить операцию поиска. Если нужная страница в ОП отсутствует, производится обращение к основной таблице на диске и после загрузки страницы в ОП корректируется и инвертированная таблица.

17.3 Сегментно-страничная организация памяти

При страничной организации предполагается, что виртуальная память — это непрерывный массив со сквозной нумерацией слов, что не всегда можно признать оптимальным. Обычно программа состоит из нескольких частей — кодовой, информационной и стековой. Так как заранее неизвестны длины этих составляющих, то удобно, чтобы при программировании каждая из них имела собственную нумерацию слов, отсчитываемых с нуля. Для этого организуют систему *сегментированной памяти,* выделяя в виртуальном пространстве независимые линейные пространства переменной длины, называемые *сегментами.* Каждый сегмент представляет собой отдельную логическую единицу информации, содержащую совокупность данных или программный код и расположенную в адресном пространстве пользователя. В каждом сегменте устанавливается своя собственная нумерация слов, начиная с нуля. Виртуальная память также разбивается на сегменты, с независимой адресацией слов внутри сегмента. Каждой составляющей программы выделяется сегмент памяти. Виртуальный адрес определяется номером сегмента и адресом внутри сегмента. Для преобразования виртуального адреса в физический используется специальная *сегментная таблица.*

Недостатком такого подхода является то, что неодинаковый размер сегментов приводит к неэффективному использованию ОП. Так, если ОП заполнена, то при замещении одного из сегментов требуется вытеснить такой, размер которого равен или больше размера нового. При многократном повторе подобных действии в ОП остается множество свободных участков, недостаточных по размеру для загрузки полного сегмента. Решением проблемы служит *сегментно-страничная организация памяти.* В ней размер сегмента выбирается не произвольно, а задается кратным размеру страницы. Сегмент может содержать то или иное, но обязательно целое число страниц, даже если одна из страниц заполнена частично. Возникает определенная иерархия в организации доступа к данным, состоящая из трех ступеней: сегмент > страница > слово. Этой структуре соответствует иерархия таблица служащих для перевода виртуальных адресов в физические. В сегментной таблице программы перечисляются все сегменты данной программы с

указанием начальных адресов СТ, относящихся к каждому сегменту. Количество страничных таблиц равно числу сегментов и любая из них определяет расположение каждой из страниц сегмента в памяти, которые могут располагаться не подряд — часть страниц может находиться в ОП, остальные — во внешней памяти. Структуру виртуального адреса и процесс преобразования его в физический адрес иллюстрирует рис. 17.3.1.

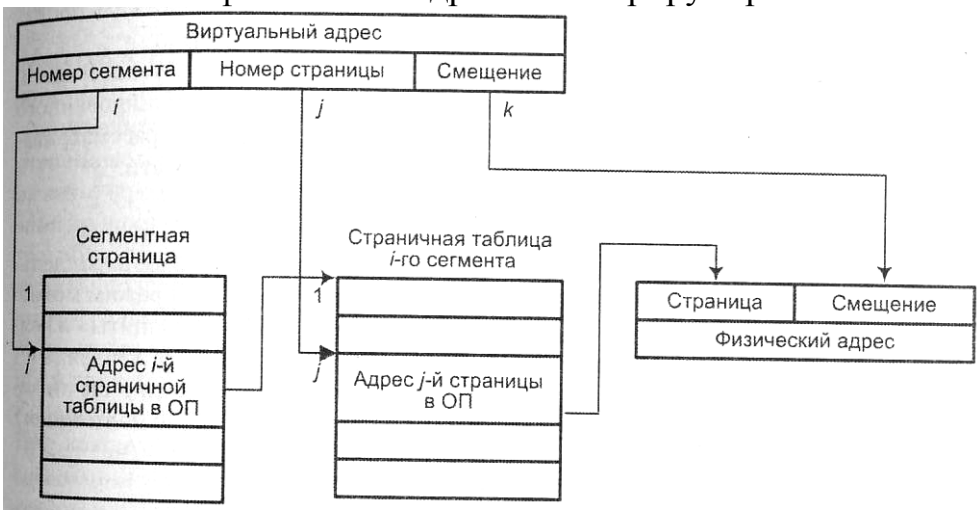

Рис. 17.3.1 Преобразование адреса при сегментно-страничной организации памяти

Для получения физического адреса необходим доступ к сегментной и одной из страничных таблиц, поэтому преобразование адреса может занимать много времени.

17.4 Организация защиты памяти

Современные вычислительные машины, как правило, работают в многопользовательском и многозадачном режимах, когда в основной памяти одновременно находятся программы, относящиеся как к разным пользователям, так и к различным задачам одного пользователя. Если даже ВМ выполняет только одну программу, в ОП, помимо этой программы и относящихся к ней данных, всегда присутствуют Фрагменты операционной системы. Каждой задаче в основной памяти выделяется свое адресное пространство. Такие пространства, если только это специально не предусмотрено, обычно независимы. В то же время в программах могут содержаться ошибки, приводящие к вторжению в адресное пространство других задач. Следствием этих ошибок может стать искажение информации, принадлежащей другим Программам. Следовательно, в ВМ обязательно должны быть предусмотрены меры, предотвращающие несанкционированное воздействие программ одного пользователя на работу программ других пользователей и на операционную систему. Особенно опасны последствия таких ошибок при нарушении адресного пространства операционной системы.

Чтобы воспрепятствовать разрушению одних программ другими, достаточно защитить область памяти данной программы от попыток записи в него со стороны других программ (защита от записи). В ряде случаев

необходимо иметь возможность защиты и от чтения со стороны других программ, например при ограничениях на доступ к системной информации.

Защита от вторжения программ в чужие адресные пространства реализуется различными средствами и способами, но в любом варианте к системе защиты предъявляются два требования: ее реализация не должна заметно снижать производительность ВМ и требовать слишком больших аппаратных затрат.

Задача обычно решается за счет сочетания программных и аппаратных средств, хотя ответственность за охрану адресных пространств от несанкционированного доступа обычно возлагается на операционную систему. В учебнике рассматриваются, главным образом, аппаратные аспекты проблемы защиты памяти.

17.5 Защита отдельных ячеек памяти

Этим видом защиты обычно пользуются при отладке новых программ параллельно с функционированием других программ. Реализовать подобный режим можно за счет выделения в каждой ячейке памяти специального «разряда защиты» и связывания его со схемой управления записью в память. Установка этого разряда в 1 блокирует запись в данную ячейку. Подобный режим использовался в вычислительных машинах предыдущих поколений (для современных ВМ он не типичен).

17.6 Кольца защиты

Защиту адресного пространства операционной системы от несанкционированного вторжения со стороны пользовательских программ обычно организуют за счет аппаратно реализованного разделения системного и пользовательского уровней привилегий. Предусматриваются как минимум два режима работы процессора: системный (режим супервизора — «надзирателя») и пользовательский. Такую структуру принято называть *кольцами защиты* и изображать в виде концентрических окружностей, где пользовательский режим представлен внешним кольцом, а системный — внутренней окружностью. В системном режиме программе доступны все ресурсы ВМ, а возможности пользовательского режима существенно ограничены. Переключение из пользовательского режима в системный осуществляется специальной командой. В большинстве современных ВМ число уровней привилегий (колец защиты) увеличено. Так, в микропроцессорах класса Pentium предусмотрено четыре уровня привилегий.

17.7 Метод граничных регистров

Данный вид защиты наиболее распространен. Метод предполагает наличие в процессоре двух *граничных регистров,* содержимое которых определяет нижнюю и верхнюю границы области памяти, куда программа имеет право доступа. Заполнение граничных регистров производится

операционной системой при загрузке программы. При каждом обращении к памяти проверяется, попадает ли используемый адрес в установленные границы. Такую проверку, например, можно организовать на этапе преобразования виртуального адреса в физический. При нарушении границы доступ к памяти блокируется, и формируется запрос прерывания, вызывающий соответствующую процедуру операционной системы. Нижнюю границу разрешенной области памяти определяет сегментный регистр. Верхняя граница подсчитывается операционной системой в соответствии с размером размещаемого в ОП сегмента.

В рассмотренной схеме необходимо, чтобы в ВМ поддерживались два режима работы: привилегированный и пользовательский. Запись информации в граничные регистры возможна лишь в привилегированном режиме.

17.8 Метод ключей защиты

Метод позволяет организовать защиту несмежных областей памяти. Память условно делится на блоки одинакового размера. Каждому блоку ставится в соответствие некоторый код, называемый *ключом защиты памяти.* Каждой программе, в свою очередь, присваивается *код защиты программы.* Условием доступа программы к конкретному блоку памяти служит совпадение ключей защиты памяти и программы, либо равенство одного из этих ключей нулю. Нулевое значение ключа защиты программы разрешает доступ ко всему адресному пространству и используется только программами операционной системы. Распределением ключей защиты программы ведает операционная система. Ключ защиты программы обычно представлен в виде отдельного поля *слова состояния программы,* хранящегося в специальном регистре. Ключи защиты памяти хранятся в специальной памяти. При каждом обращении к ОП специальная комбинационная схема производит сравнение ключей защиты памяти и программы. При совпадении доступ к памяти разрешается. Действия в случае несовпадения ключей зависят от того, какой вид доступа запрещен: при записи, при чтении или в обоих случаях. Если выяснилось, что данный вид доступа запрещен, то так же как и в методе граничных регистров формируется запрос прерывания и вызывается соответствующая процедура