**Лекция №13. Тема: Стратегии распределения памяти.**

Наиболее распространенными реализациями виртуальной памяти является страничное, сегментное и странично- сегментное распределение памяти, а также свопинг.

**Страничное распределение**

На рисунке 1 показана схема страничного распределения памяти. Виртуальное адресное пространство каждого процесса делится на части одинакового, фиксированного для данной системы размера, называемые виртуальными страницами. В общем случае размер виртуального адресного пространства не является кратным размеру страницы, поэтому последняя страница каждого процесса дополняется фиктивной областью.

Вся оперативная память машины также делится на части такого же размера, называемые физическими страницами (или блоками).

Размер страницы обычно выбирается равным степени двойки: 512, 1024 и т.д., это позволяет упростить механизм преобразования адресов.

При загрузке процесса часть его виртуальных страниц помещается в оперативную память, а остальные - на диск. Смежные виртуальные страницы не обязательно располагаются в смежных физических страницах. При загрузке операционная система создает для каждого процесса информационную структуру - таблицу страниц, в которой устанавливается соответствие между номерами виртуальных и физических страниц для страниц, загруженных в оперативную память, или делается отметка о том, что виртуальная страница выгружена на диск. Кроме того, в таблице страниц содержится управляющая информация, такая как признак модификации страницы, признак невыгружаемости (выгрузка некоторых страниц может быть запрещена), признак обращения к странице (используется для подсчета числа обращений за определенный период времени) и другие данные, формируемые и используемые механизмом виртуальной памяти.

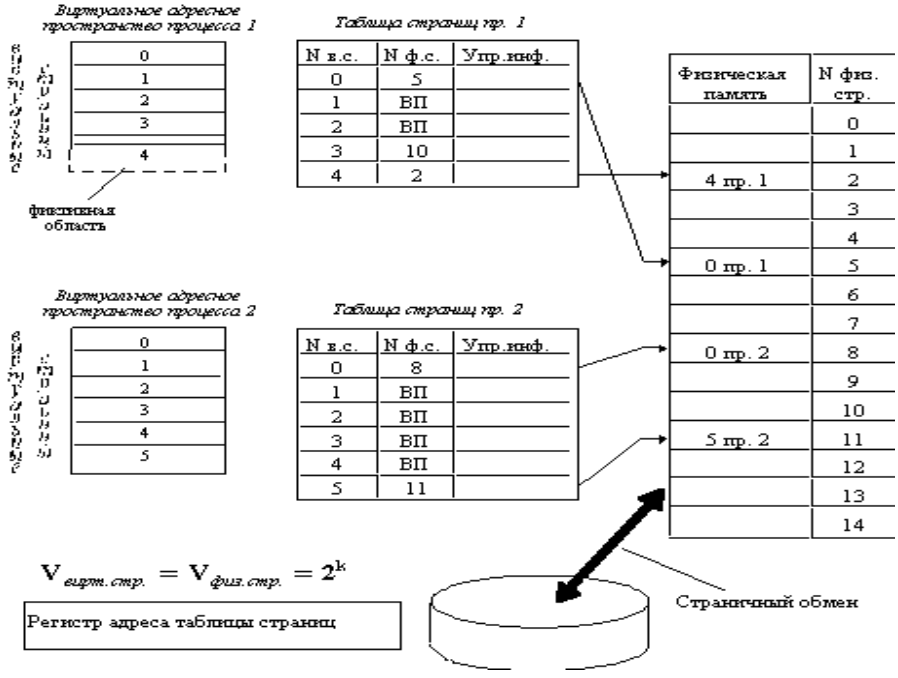



*Рис. 1. Страничное распределение памяти*

При активизации очередного процесса в специальный регистр процессора загружается адрес таблицы страниц данного процесса.

При каждом обращении к памяти происходит чтение из таблицы страниц информации о виртуальной странице, к которой произошло обращение. Если данная виртуальная страница находится в оперативной памяти, то выполняется преобразование виртуального адреса в физический. 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 готовых. Параллельно программа обработки страничного прерывания находит на диске требуемую виртуальную страницу и пытается загрузить ее в оперативную память. Если в памяти имеется свободная физическая страница, то загрузка выполняется немедленно, если же свободных страниц нет, то решается вопрос, какую страницу следует выгрузить из оперативной памяти.

В данной ситуации может быть использовано много разных критериев выбора, наиболее популярные из них следующие:

- дольше всего не использовавшаяся страница,
- первая попавшаяся страница,
- страница, к которой в последнее время было меньше всего обращений.

В некоторых системах используется понятие рабочего множества страниц. Рабочее множество определяется для каждого процесса и представляет собой перечень наиболее часто используемых страниц, которые должны постоянно находиться в оперативной памяти и поэтому не подлежат выгрузке.

После того, как выбрана страница, которая должна покинуть оперативную память, анализируется ее признак модификации (из таблицы страниц). Если выталкиваемая страница с момента загрузки была модифицирована, то ее новая версия должна быть переписана на диск. Если нет, то она может быть просто уничтожена, то есть соответствующая физическая страница объявляется свободной.

Рассмотрим механизм преобразования виртуального адреса в физический при страничной организации памяти (рисунок 2.13).

Виртуальный адрес при страничном распределении может быть представлен в виде пары (p, s), где p - номер виртуальной страницы процесса (нумерация страниц начинается с 0), а s - смещение в пределах виртуальной страницы. Учитывая, что размер страницы равен 2 в степени к, смещение s может быть получено простым отделением k младших разрядов в двоичной записи виртуального адреса. Оставшиеся старшие разряды представляют собой двоичную запись номера страницы p.


*Рис. 2. Механизм преобразования виртуального адреса в физический при страничной организации памяти*

При каждом обращении к оперативной памяти аппаратными средствами выполняются следующие действия:

1. на основании начального адреса таблицы страниц (содержимое регистра адреса таблицы страниц), номера виртуальной страницы (старшие разряды виртуального адреса) и длины записи в таблице страниц (системная константа) определяется адрес нужной записи в таблице,
2. из этой записи извлекается номер физической страницы,
3. к номеру физической страницы присоединяется смещение (младшие разряды виртуального адреса).

Использование в пункте (3) того факта, что размер страницы равен степени 2, позволяет применить операцию конкатенации (присоединения) вместо более длительной операции сложения, что уменьшает время получения физического адреса, а значит повышает производительность компьютера.

На производительность системы со страничной организацией памяти влияют временные затраты, связанные с обработкой страничных прерываний и преобразованием виртуального адреса в физический. При часто возникающих страничных прерываниях система может тратить большую часть времени впустую, на свопинг страниц. Чтобы уменьшить частоту страничных прерываний, следовало бы увеличивать размер страницы. Кроме того, увеличение размера страницы уменьшает размер таблицы страниц, а значит уменьшает затраты памяти. С другой стороны, если страница велика, значит велика и фиктивная область в последней виртуальной странице каждой программы. В среднем на каждой программе теряется половина объема страницы, что в сумме при большой странице может составить существенную величину. Время преобразования виртуального адреса в физический в значительной степени определяется временем доступа к таблице страниц. В связи с этим таблицу страниц стремятся размещать в "быстрых" запоминающих устройствах. Это может

быть, например, набор специальных регистров или память, использующая для уменьшения времени доступа ассоциативный поиск и кэширование данных.

Страничное распределение памяти может быть реализовано в упрощенном варианте, без выгрузки страниц на диск. В этом случае все виртуальные страницы всех процессов постоянно находятся в оперативной памяти. Такой вариант страничной организации хотя и не предоставляет пользователю виртуальной памяти, но почти исключает фрагментацию за счет того, что программа может загружаться в несмежные области, а также того, что при загрузке виртуальных страниц никогда не образуется остатков.

**Сегментное распределение**

При страничной организации виртуальное адресное пространство процесса делится механически на равные части. Это не позволяет дифференцировать способы доступа к разным частям программы (сегментам), а это свойство часто бывает очень полезным. Например, можно запретить обращаться с операциями записи и чтения в кодовый сегмент программы, а для сегмента данных разрешить только чтение. Кроме того, разбиение программы на "осмысленные" части делает принципиально возможным разделение одного сегмента несколькими процессами. Например, если два процесса используют одну и ту же математическую подпрограмму, то в оперативную память может быть загружена только одна копия этой подпрограммы.

Рассмотрим, каким образом сегментное распределение памяти реализует эти возможности (рисунок 2). Виртуальное адресное пространство процесса делится на сегменты, размер которых определяется программистом с учетом смыслового значения содержащейся в них информации. Отдельный сегмент может представлять собой подпрограмму, массив данных и т.п. Иногда сегментация программы выполняется по умолчанию компилятором.

При загрузке процесса часть сегментов помещается в оперативную память (при этом для каждого из этих сегментов операционная система подыскивает подходящий участок свободной памяти), а часть сегментов размещается в дисковой памяти. Сегменты одной программы могут занимать в оперативной памяти несмежные участки. Во время загрузки система создает таблицу сегментов процесса (аналогичную таблице страниц), в которой для каждого сегмента указывается начальный физический адрес сегмента в оперативной памяти, размер сегмента, правила доступа, признак модификации, признак обращения к данному сегменту за последний интервал времени и 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.

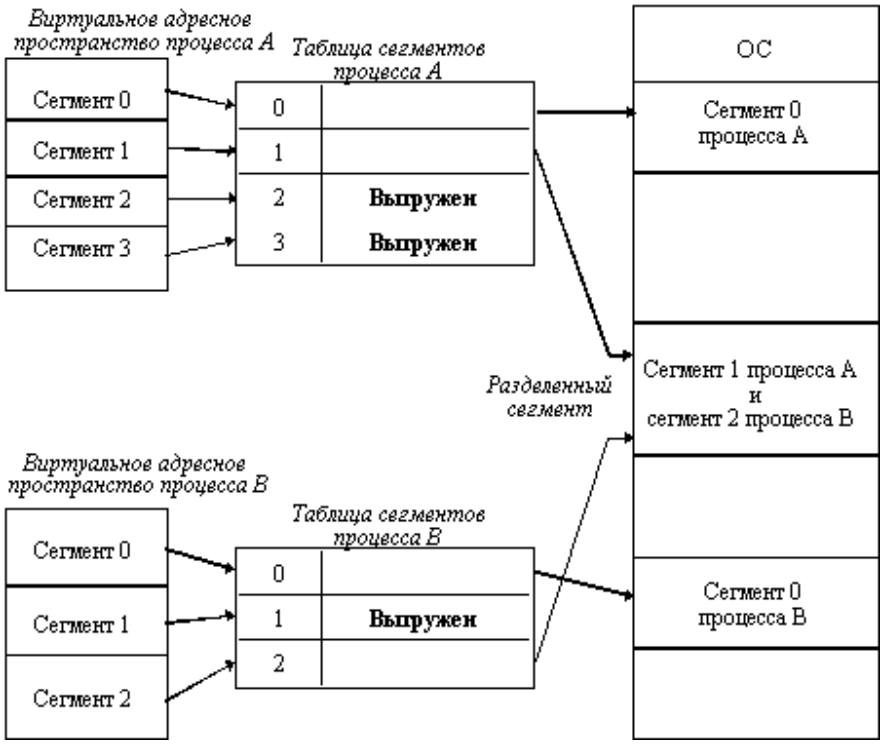



*Рис. 2.14. Распределение памяти сегментами*

Система с сегментной организацией функционирует аналогично системе со страничной организацией: время от времени происходят прерывания, связанные с отсутствием нужных сегментов в памяти, при необходимости освобождения памяти некоторые сегменты выгружаются, при каждом обращении к оперативной памяти выполняется преобразование виртуального адреса в физический. Кроме того, при обращении к памяти проверяется, разрешен ли доступ требуемого типа к данному сегменту.

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 начального физического адреса сегмента, найденного в таблице сегментов по номеру g, и смещения s.

Недостатком данного метода распределения памяти является фрагментация на уровне сегментов и более медленное по сравнению со страничной организацией преобразование адреса.

**Странично-сегментное распределение**

Как видно из названия, данный метод представляет собой комбинацию страничного и сегментного распределения памяти и, вследствие этого, сочетает в себе достоинства обоих подходов. Виртуальное пространство процесса делится на сегменты, а каждый сегмент в свою очередь делится на виртуальные страницы, которые нумеруются в пределах сегмента. Оперативная память делится на физические страницы. Загрузка процесса выполняется операционной системой постранично, при этом часть страниц размещается в оперативной памяти, а часть на диске. Для каждого сегмента создается своя таблица страниц, структура

которой полностью совпадает со структурой таблицы страниц, используемой при страничном распределении. Для каждого процесса создается таблица сегментов, в которой указываются адреса таблиц страниц для всех сегментов данного процесса. Адрес таблицы сегментов загружается в специальный регистр процессора, когда активизируется соответствующий процесс. На рисунке 2.15 показана схема преобразования виртуального адреса в физический для данного метода.

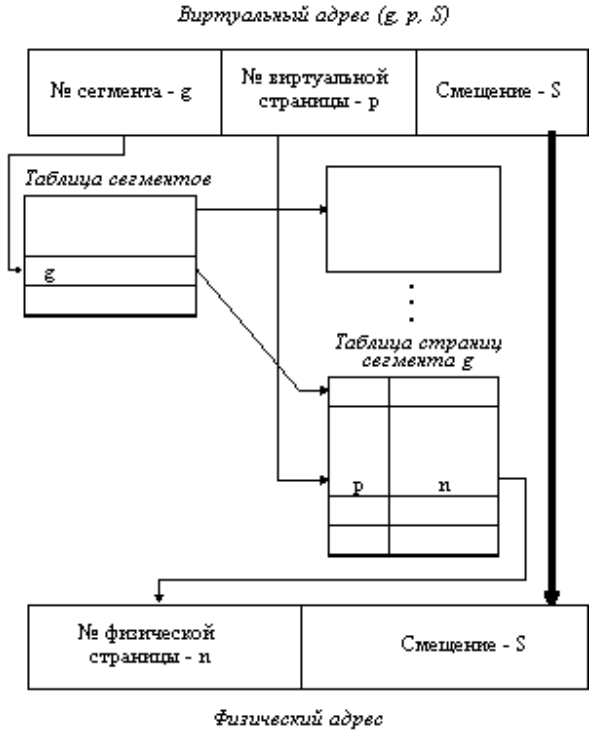



*Рис. 3. Схема преобразования виртуального адреса в физический для сегментно-страничной организации памяти*

**Свопинг**

Разновидностью виртуальной памяти является свопинг.

На рисунке 3 показан график зависимости коэффициента загрузки процессора в зависимости от числа одновременно выполняемых процессов и доли времени, проводимого этими процессами в состоянии ожидания ввода-вывода.

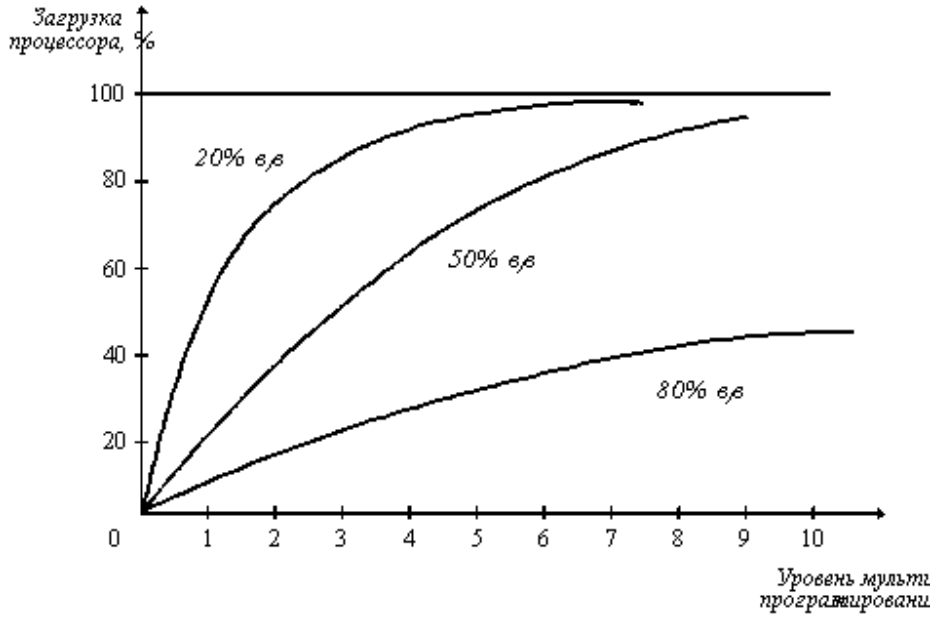



*Рис. 4. Зависимость загрузки процессора от числа задач и интенсивности ввода-вывода*

Из рисунка видно, что для загрузки процессора на 90% достаточно всего трех счетных задач. Однако для того, чтобы обеспечить такую же загрузку интерактивными задачами, выполняющими интенсивный ввод-вывод, потребуются десятки таких задач. Необходимым условием для выполнения задачи является загрузка ее в оперативную память, объем которой ограничен. В этих условиях был предложен метод организации вычислительного процесса, называемый свопингом. В соответствии с этим методом некоторые процессы (обычно находящиеся в состоянии ожидания) временно выгружаются на диск. Планировщик операционной системы не исключает их из своего рассмотрения, и при наступлении условий активизации некоторого процесса, находящегося в области свопинга на диске, этот процесс перемещается в оперативную память. Если свободного места в оперативной памяти не хватает, то выгружается другой процесс.

При свопинге, в отличие от рассмотренных ранее методов реализации виртуальной памяти, процесс перемещается между памятью и диском целиком, то есть в течение некоторого времени процесс может полностью отсутствовать в оперативной памяти. Существуют различные алгоритмы выбора процессов на загрузку и выгрузку, а также различные способы выделения оперативной и дисковой памяти загружаемому процессу.

**Иерархия запоминающих устройств. Принцип кэширования данных**

Память вычислительной машины представляет собой иерархию запоминающих устройств (внутренние регистры процессора, различные типы сверхоперативной и оперативной памяти, диски, ленты), отличающихся средним временем доступа и стоимостью хранения данных в расчете на один бит (рисунок 5). Пользователю хотелось бы иметь и недорогую и быструю память. Кэш-память представляет некоторое компромиссное решение этой проблемы.

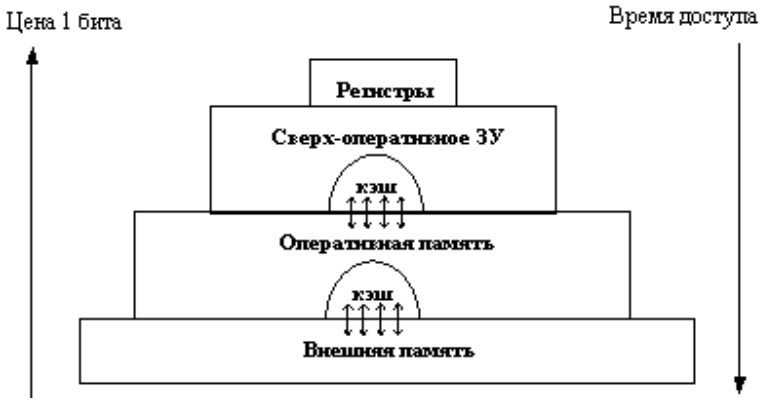
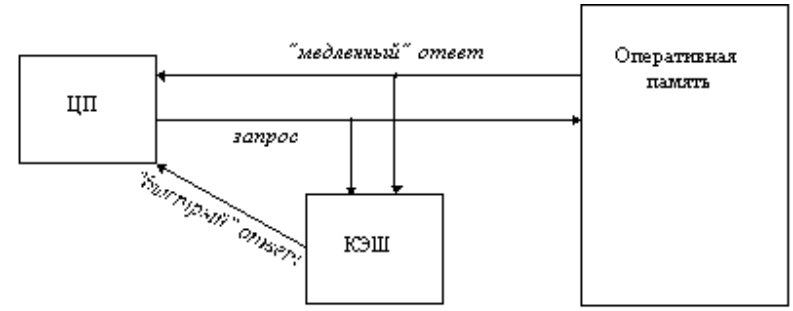



*Рис. 5. Иерархия ЗУ*

*Кэш-память* - это способ организации совместного функционирования двух типов запоминающих устройств, отличающихся временем доступа и стоимостью хранения данных, который позволяет уменьшить среднее время доступа к данным за счет динамического копирования в "быстрое" ЗУ наиболее часто используемой информации из "медленного" ЗУ.

Кэш-памятью часто называют не только способ организации работы двух типов запоминающих устройств, но и одно из устройств - "быстрое" ЗУ. Оно стоит дороже и, как правило, имеет сравнительно небольшой объем. Важно, что механизм кэш-памяти является прозрачным для пользователя, который не должен сообщать никакой информации об интенсивности использования данных и не должен никак участвовать в перемещении данных из ЗУ одного типа в ЗУ другого типа, все это делается автоматически системными средствами.

Рассмотрим частный случай использования кэш-памяти для уменьшения среднего времени доступа к данным, хранящимся в оперативной памяти. Для этого между процессором и оперативной памятью помещается быстрое ЗУ, называемое просто кэш-памятью (рисунок 2.18). В качестве такового может быть использована, например, ассоциативная память. Содержимое кэш-памяти представляет собой совокупность записей обо всех загруженных в нее элементах данных. Каждая запись об элементе данных включает в себя адрес, который этот элемент данных имеет в оперативной памяти, и управляющую информацию: признак модификации и признак обращения к данным за некоторый последний период времени.




Структура кэш-памяти

| Адрес данных в ОП | Данные | Управл. информация | |
|---|---|---|---|
| | | Бит модиф. | Бит обращ. |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

*Рис. 6. Кэш-память*

В системах, оснащенных кэш-памятью, каждый запрос к оперативной памяти выполняется в соответствии со следующим алгоритмом:

1. Просматривается содержимое кэш-памяти с целью определения, не находятся ли нужные данные в кэш-памяти; кэш-память не является адресуемой, поэтому поиск нужных данных осуществляется по содержимому - значению поля "адрес в оперативной памяти", взятому из запроса.
2. Если данные обнаруживаются в кэш-памяти, то они считываются из нее, и результат передается в процессор.
3. Если нужных данных нет, то они вместе со своим адресом копируются из оперативной памяти в кэш-память, и результат выполнения запроса передается в процессор. При копировании данных может оказаться, что в кэш-памяти нет свободного места, тогда выбираются данные, к которым в последний период было меньше всего обращений, для вытеснения из кэш-памяти. Если вытесняемые данные были модифицированы за время нахождения в кэш-памяти, то они переписываются в оперативную память. Если же эти данные не были модифицированы, то их место в кэш-памяти объявляется свободным.

На практике в кэш-память считывается не один элемент данных, к которому произошло обращение, а целый блок данных, это увеличивает вероятность так называемого "попадания в кэш", то есть нахождения нужных данных в кэш-памяти.

Покажем, как среднее время доступа к данным зависит от вероятности попадания в кэш. Пусть имеется основное запоминающие устройство со средним временем доступа к данным t1 и кэш-память, имеющая время доступа t2, очевидно, что t2<t1. Обозначим через t среднее время доступа к данным в системе с кэш-памятью, а через p -вероятность попадания в кэш. По формуле полной вероятности имеем:

$$t = t1((1 - p) + t2(p$$

Из нее видно, что среднее время доступа к данным в системе с кэш-памятью линейно зависит от вероятности попадания в кэш и изменяется от среднего времени доступа в основное ЗУ (при p=0) до среднего времени доступа непосредственно в кэш-память (при p=1).

В реальных системах вероятность попадания в кэш составляет примерно 0,9. Высокое значение вероятности нахождения данных в кэш-памяти связано с наличием у данных объективных свойств: пространственной и временной локальности.

- *Пространственная локальность*. Если произошло обращение по некоторому адресу, то с высокой степенью вероятности в ближайшее время произойдет обращение к соседним адресам.
- *Временная локальность*. Если произошло обращение по некоторому адресу, то следующее обращение по этому же адресу с большой вероятностью произойдет в ближайшее время.

**Все предыдущие рассуждения справедливы и для других пар запоминающих устройств, например, для оперативной памяти и внешней памяти. В этом случае уменьшается среднее время доступа к данным, расположенным на диске, и роль кэш-памяти выполняет буфер в оперативной памяти. Средства аппаратной поддержки управления памятью и многозадачной среды в микропроцессорах Intel 80386, 80486 и Pentium**

Процессоры Intel 80386, 80486 и Pentium с точки зрения рассматриваемых в данном разделе вопросов имеют аналогичные средства, поэтому для краткости в тексте используется термин "процессор i386", хотя вся информация этого раздела в равной степени относится к трем моделям процессоров фирмы Intel.

Процессор i386 имеет два режима работы - реальный (real mode) и защищенный (protected mode). В реальном режиме процессор i386 работает как быстрый процессор 8086 с несколько расширенным набором команд. В защищенном режиме процессор i386 может использовать все механизмы 32-х разрядной организации памяти, в том числе механизмы поддержки виртуальной памяти и механизмы переключения задач. Кроме этого, в защищенном режиме для каждой задачи процессор i386 может эмулировать 86 и 286 процессоры, которые в этом случае называются виртуальными процессорами. Таким образом, при многозадачной работе в защищенном режиме процессор i386 работает как несколько виртуальных процессоров, имеющих общую память. В отличие от реального режима, режим виртуального процессора i86, который называется в этом случае режимом V86, поддерживает страничную организацию памяти и средства многозадачности. Поэтому задачи, выполняющиеся в режиме V86, используют те же средства межзадачной защиты и защиты ОС от пользовательских задач, что и задачи, работающие в защищенном режиме i386. Однако максимальный размер виртуального адресного пространства составляет 1 Мб, как и у процессора i86.

Переключение процессора i386 из реального режима в защищенный и обратно осуществляется просто путем выполнения команды MOV, которая изменяет бит режима в одном из управляющих регистров процессора. Переход процессора в режим V86 происходит похожим образом путем изменения значения определенного бита в другом регистре процессора.

**Средства поддержки сегментации памяти**

Физическое адресное пространство процессора i386 составляет 4 Гбайта, что определяется 32-разрядной шиной адреса. Физическая память является линейной с адресами от 00000000 до FFFFFFFF в шестнадцатеричном представлении. Виртуальный адрес, используемый в программе, представляет собой пару - номер сегмента и смещение внутри сегмента. Смещение хранится в соответствующем поле команды, а номер сегмента - в одном из шести сегментных регистров процессора (CS, SS, DS, ES, FS или GS), каждый из которых является 16-битным. Средства сегментации образуют верхний уровень средств управления виртуальной памятью процессора i386, а средства страничной организации - нижний уровень. Средства страничной организации могут быть как включены, так и выключены (за счет установки определенного бита в управляющем регистре процессора), и в зависимости от этого изменяется смысл преобразования виртуального адреса, которое выполняют средства сегментации. Сначала рассмотрим случай работы средств сегментации при отключенном механизме управления страницами.

Устройство выгрузки является устройством блочного типа, которое представляет собой конфигурируемый раздел диска. Тогда как обычно ядро выделяет место для файлов по одному блоку за одну операцию, на устройстве выгрузки пространство выделяется группами смежных блоков. Пространство, выделяемое для файлов, используется статическим образом; поскольку схема назначения пространства под файлы действует в течение длительного периода времени, ее гибкость понимается в смысле сокращения числа случаев фрагментации и, следовательно, объемов неиспользуемого пространства в файловой системе. Выделение пространства на устройстве выгрузки, напротив, является временным, в сильной степени зависящим от механизма диспетчеризации процессов. Процесс, размещаемый на устройстве выгрузки, в конечном итоге вернется в основную память, освобождая место на внешнем устройстве. Поскольку время является решающим фактором и с учетом того, что ввод-вывод данных за одну мультиблочную операцию происходит быстрее, чем за несколько одноблочных операций, ядро выделяет на устройстве выгрузки непрерывное пространство, не беря во внимание возможную фрагментацию.
Так как схема выделения пространства на устройстве выгрузки отличается от схемы, используемой для файловых систем, структуры данных, регистрирующие свободное пространство, должны также отличаться. Пространство, свободное в файловых системах, описывается с помощью связного списка свободных блоков, доступ к которому осуществляется через суперблок файловой системы, информация о свободном пространстве на устройстве выгрузки собирается в таблицу, именуемую "карта памяти устройства". Карты памяти, помимо устройства выгрузки, используются и другими системными ресурсами (например, драйверами некоторых устройств), они дают возможность распределять память устройства (в виде смежных блоков) по методу первого подходящего.
Каждая строка в карте памяти состоит из адреса распределяемого ресурса и количества доступных единиц ресурса; ядро интерпретирует элементы строки в соответствии с типом карты. В самом начале карта памяти состоит из одной строки, содержащей адрес и общее количество ресурсов. Если карта описывает распределение памяти на устройстве выгрузки, ядро трактует каждую единицу ресурса как группу дисковых блоков, а адрес - как смещение в блоках от начала области выгрузки. Первоначальный вид карты памяти для устройства выгрузки, состоящего из 10000 блоков с начальным адресом, равным 1, показан. Выделяя и освобождая ресурсы, ядро корректирует карту памяти, заботясь о том, чтобы в ней постоянно содержалась точная информация о свободных ресурсах в системе. Процесс подкачки не может стронуть с места ни один процесс в течение первых двух секунд, поскольку этого

требует условие нахождения перемещаемого процесса в течение этого интервала на одном месте (в памяти или на устройстве выгрузки), однако по истечении 2 секунд процесс подкачки выгружает процессы A и B и загружает на их место процессы C и D. Он пытается также загрузить и процесс E, но терпит неудачу, поскольку в основной памяти недостаточно места для этого. На 3-секундной отметке процесс E все еще годен для загрузки, поскольку он находился все 3 секунды на устройстве выгрузки, но процесс подкачки не может выгрузить из памяти ни один из процессов, ибо они находятся в памяти менее 2 секунд. На 4-секундной отметке процесс подкачки выгружает процессы C и D и загружает вместо них процессы E и A.

Процесс подкачки выбирает процессы для загрузки, основываясь на продолжительности их пребывания на устройстве выгрузки. В качестве другого критерия может применяться более высокий приоритет загружаемого процесса по сравнению с остальными, готовыми к выполнению процессами, поскольку такой процесс более предпочтителен для запуска. Практика показала, что такой подход "несколько" повышает пропускную способность системы в условиях сильной загруженности.