Виртуальная память

- Оперативная память, с которой работают вычислительная система и программное обеспечение, которая не реализована физически
- Способ организации оперативной памяти, отличный от ее физической организации

Адресное пространство

- Физическое адресное пространство – совокупность адресов физической оперативной памяти
- Допустимое адресное пространство – множество адресов ОП, определенное разрядностью шины адреса и другими особенностями аппаратной платформы
- Виртуальное адресное пространство – совокупность адресов виртуальной памяти, формат и множество которых определены в соответствии с размерами и способом организации виртуальной памяти

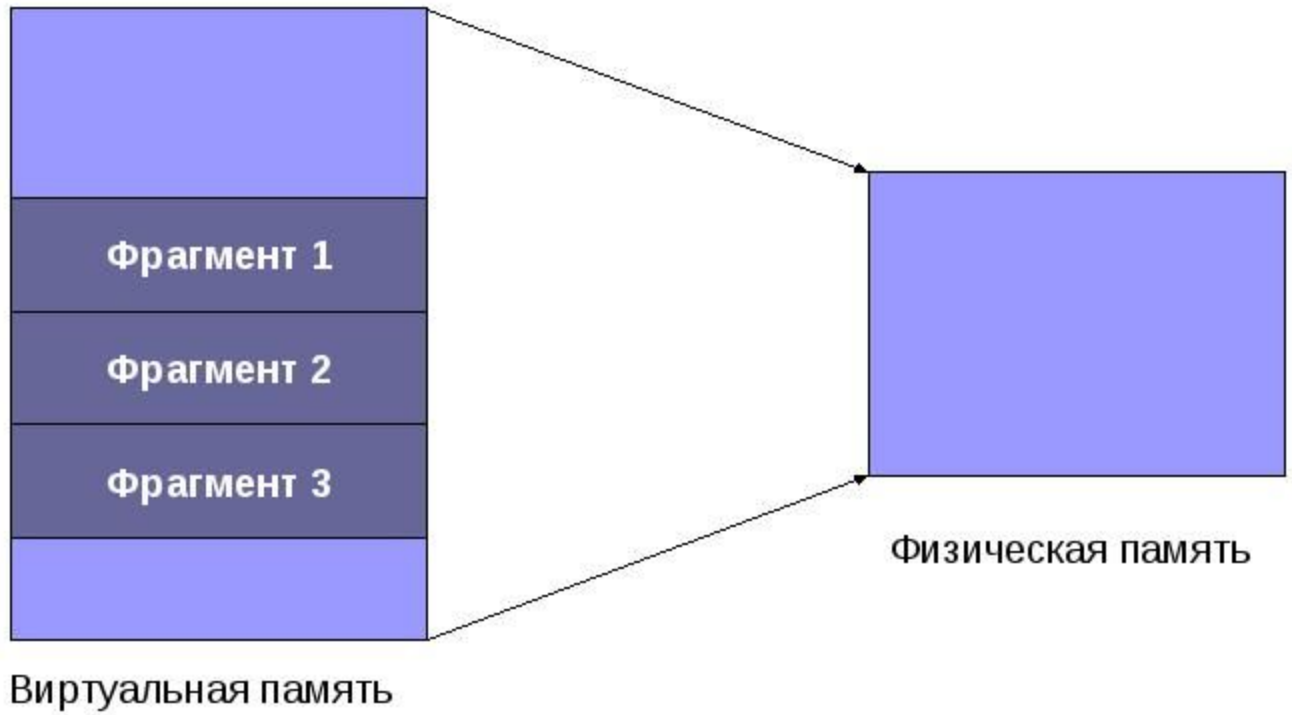
Использование виртуального адресного пространства

- На аппаратном уровне – для оптимизации управления физической памятью (сегментация памяти в MS DOS)
- На уровне операционной системы:
  - для оптимизации управления памятью как ресурсом системы;
  - для увеличения объема доступного адресного пространства (за счет свопинга);
  - для обеспечения переносимости программ (обеспечение различных моделей выделения ОП под процессы различных ОС)
- На уровне программ:
  - для обеспечения независимости программ от их размещения в памяти;
  - для увеличения допустимых объемов программ

Отображение виртуального адресного пространства

- Совокупность правил, которые определяют, как виртуальный адрес будет преобразован в физический (реальный) адрес
- Схема преобразования виртуальных адресов в физические
- Установка соответствия областей виртуального адресного пространства областям физической памяти

Отображение виртуального адресного пространства



Отображение виртуального адресного пространства как схема преобразования адресов

- Физический адрес всегда является *линейным*, т.е. это номер байта в физической памяти от начала памяти
  - Минимальный физический адрес всегда равен 0
  - Максимальный физический адрес всегда определяется конкретной реализацией аппаратной памяти (если объем ФОП составляет 1 Гбайт, то максимальный физический адрес будет равен 2^{30}-1 или 1.073.741.823)

Отображение виртуального адресного пространства как схема преобразования адресов

- Линейный адрес – это адрес в допустимом адресном пространстве, которое так же считается линейным.
  - Минимальный линейный адрес равен 0
  - Максимальный линейный адрес определен разрядностью шины адреса
    - Для Pentium III – 2^{32}-1 или 4.294.967.295
    - Для Pentium IV – 2^{36}-1 или 64.719.476.735
    - Для 64^{x}-разрядных процессоров – 2^{64} или 18.446.744.073.709.599.999

Отображение виртуального адресного пространства как схема преобразования адресов

- Виртуальный адрес может иметь любую структуру
- Структура виртуального адреса определяется способом организации виртуальной памяти
  - Линейный
  - Сегментный
  - Страничный
  - Сегментно-страничный
- Способ организации памяти определяет
  - Схему преобразования ВА в ФА (или хотя бы в ЛА) на аппаратном уровне и уровне ОС
  - Представление программ в памяти и формирование внутренних виртуальных адресов в программах

Способы организации памяти

- Линейный
- Сегментный
- Страничный
- Сегментно-страничный

Линейный способ организации памяти

- ВАП считается линейным
- Допустимые значения адресов определяются
  - Разрядностью шины адреса
  - Особенностями аппаратной платформы (процессором)
  - Особенностями ОС
- Используется только на аппаратном уровне и уровне ОС
  - Сокращение накладных расходов, которые несут процессор и ОС при преобразовании виртуальных адресов в физические

Линейный способ организации памяти

- Использование на уровне программ:
  - Усложняет разработку программного обеспечения, особенно компиляцию
  - Усложняет загрузку программ (необходимо загружать программы по абсолютному адресу или перевычислять все адреса в программе в момент запуска)
  - Замедляет и усложняет работу механизма свопинга (пересчет адресов внутри программ при перемещении их в памяти или привязка программ только к определенным участкам ФАП и ВАП)

Линейный способ организации памяти

- Использовался в старых ОС
- Используется в современных вычислительных и ОС наряду с другими способами организации памяти (для упрощения управления программами)
- Поддерживается аппаратными платформами Sun UltraSPARC, Intel Itanium (IA-64), Intel Pentium D/T/M, Intel Core, AMD Athlon64, AMD Opteron

Сегментный способ организации памяти

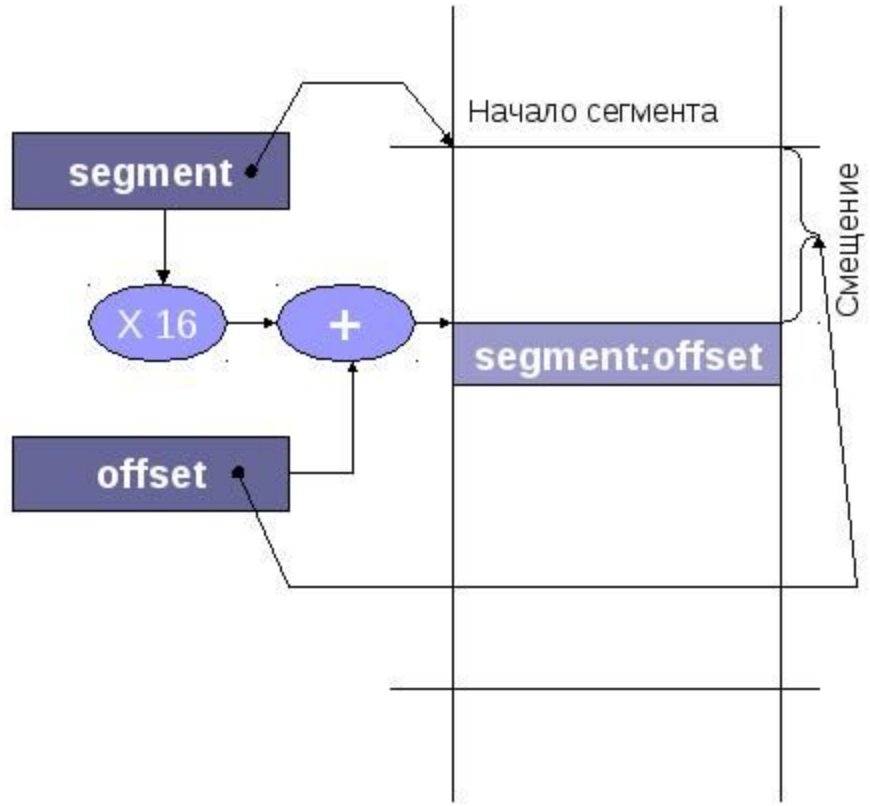
- Сегмент – это неделимый участок адресного пространства произвольного размера
- Вся память делится на сегменты
- Для обращения к ячейке памяти необходимо указать адрес начала сегмента (segment) и смещение ячейки внутри сегмента (offset)

ФА = segment + offset

- Фактически, длина сегмента ограничена разрядность смещения
  - В 16-ти разрядных системах длина сегмента 64 Кбайт
  - В 32-х разрядных системах – 4 Гбайт

Сегментный способ организации памяти: MS DOS и реальный режим работы Intel

- Адрес начала сегмента хранится в одном из сегментных регистров: CS, DS, ES, FS, GS, SS
- Смещение имеет длину 16 бит
- Максимальный размер сегмента – 64 Кбайт
- ФА = segment*16+offset
- Используется и на аппаратном уровне, и в программах



Сегментный способ организации памяти: MS DOS и реальный режим работы Intel

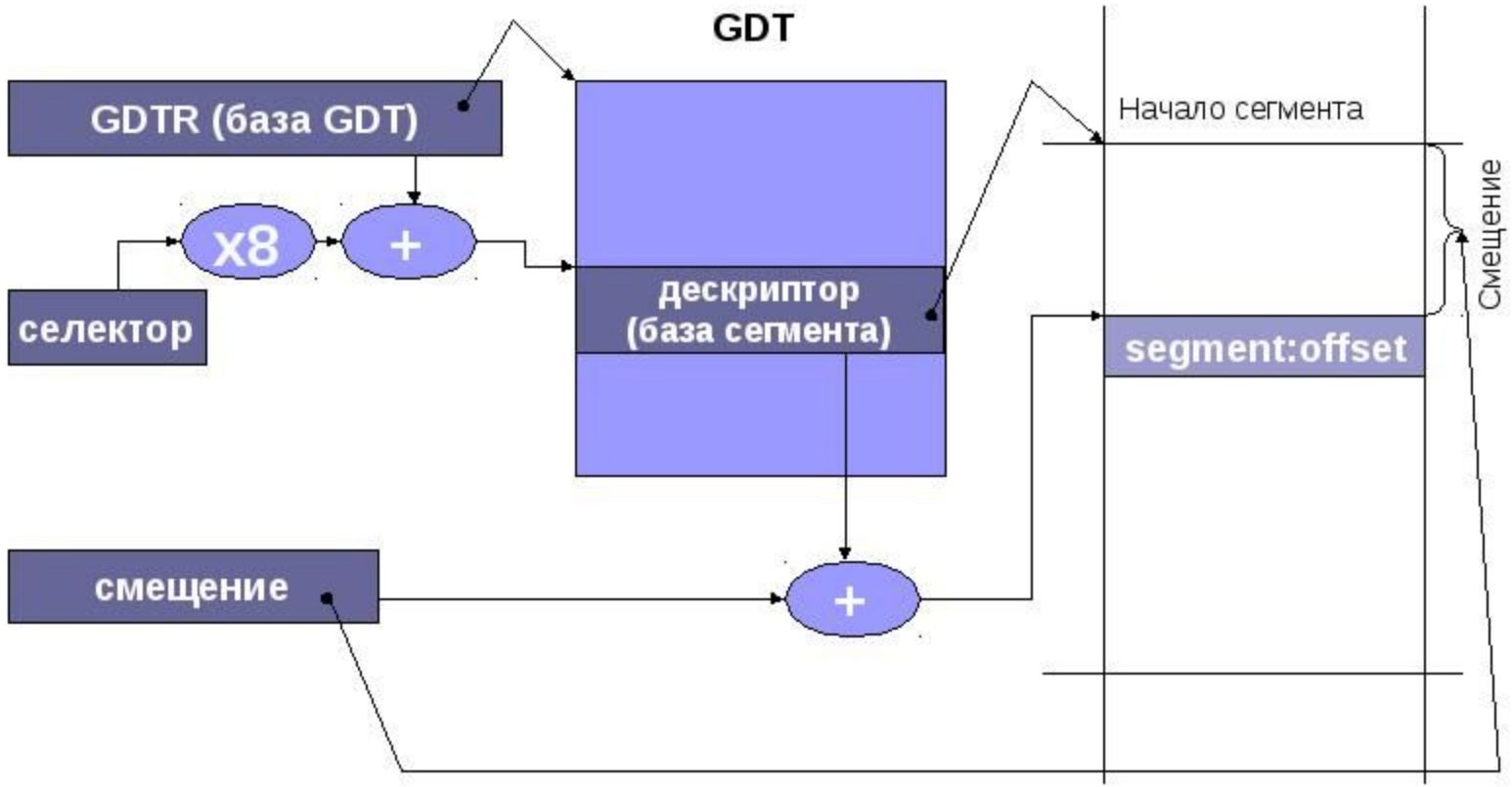
Достоинства

- Простота
- Универсальность

Недостатки

- Малый размер сегмента
- Малое количество сегментных регистров
- Малый объем ОП доступен в один момент времени

Сегментный способ организации памяти: современные ОС и защищенный режим работы Intel



Сегментный способ организации памяти: современные ОС и защищенный режим работы Intel

- Для адресации используются 32-х разрядные линейные адреса допустимого адресного пространства (базы) и 32-х разрядные смещения
- Базы сегментов хранятся в дескрипторах специальной таблицы GDT, где, кроме них, хранится и другая информация о сегментах
- Каждый дескриптор имеет длину 8 байт (4 байта база + 4 байта дополнительная информация)
- Номер дескриптора сегмента (селектор) хранится в сегментном регистре
- Номер имеет длину 13 разрядов

Сегментный способ организации памяти: современные ОС и защищенный режим работы Intel

- В GDT может находится 2^{13} (8 192) дескриптора
- Сама GDT занимает 64 Кбайта памяти
- Т.к. используются 32-х разрядные смещения, максимальный размер сегмента составляет 4 Гбайта
- Всего можно адресовать 8 192 сегмента по 4 Гбайта каждый, т.е. 32 Тбайта памяти
- Но т.к. сегментных регистров всего 6, одновременно доступно лишь 24 Гбайта памяти
- Т.к. сама вычислительная система 32-х разрядная допустимое и доступное адресное пространство должно быть отображено на первые 4 Гбайта памяти

Сегментный способ организации памяти: современные ОС и защищенный режим работы Intel

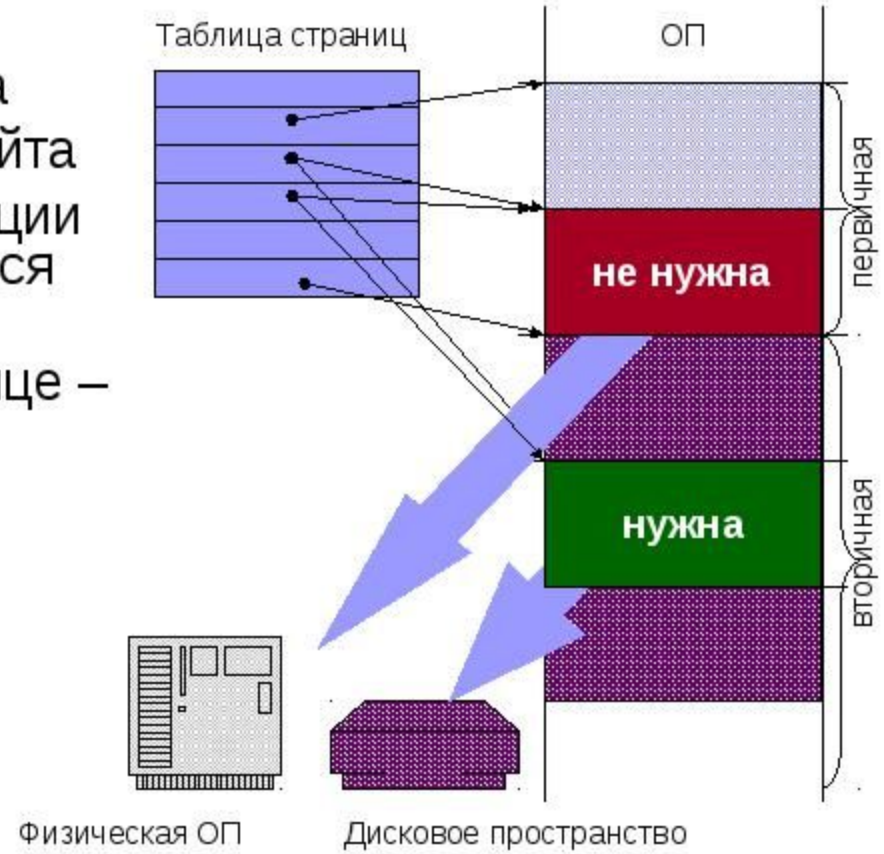
- Виртуальный адрес представляется в виде segment:offset (сегмент:смещение)
- По имени segment выбирается сегментный регистр
- Из сегментного регистра выбирается номер дескриптора нужного сегмента
- Номер умножается на 8 ® смещение дескриптора в таблице GDT
- Смещение дескриптора суммируется с базой GDT, хранящейся в регистре GDTR ® можно обратиться к дескриптору сегмента по его линейному адресу
- Из дескриптора сегмента выбирается база сегмента
- База сегмента суммируется со смещением ® **линейный адрес ячейки**

Страничный способ организации памяти

- Страница – это непрерывный участок оперативной памяти фиксированного размера
- Традиционно размер страницы считается равным 4 Кбайт
- Использование страниц позволяет проще реализовать некоторые механизмы управления памятью в ОС, например свопинг

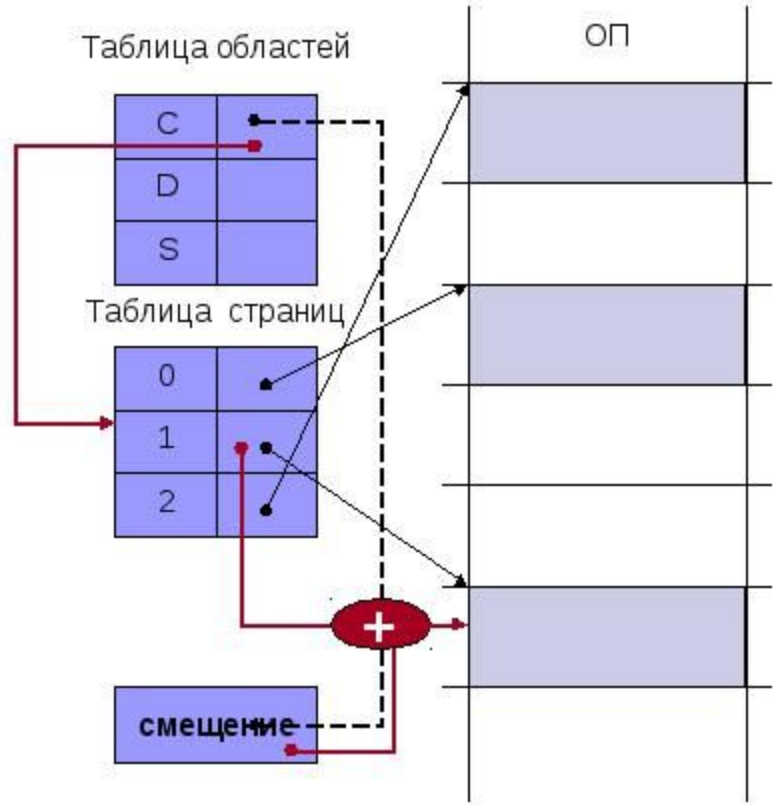
Традиционная страничная организация памяти: **VAX-11 и UNIX**

- 4 Гбайт виртуального адресного пространства
- Размер страницы 4 Кбайта
- Для хранения информации о страницах используется таблица страниц
- Размер записи о странице – 4 байта
- Старший бит – бит присутствия
  - 0 – во вторичной памяти
  - 1 – в первичной
- Аппаратная поддержка свопинга по запросу



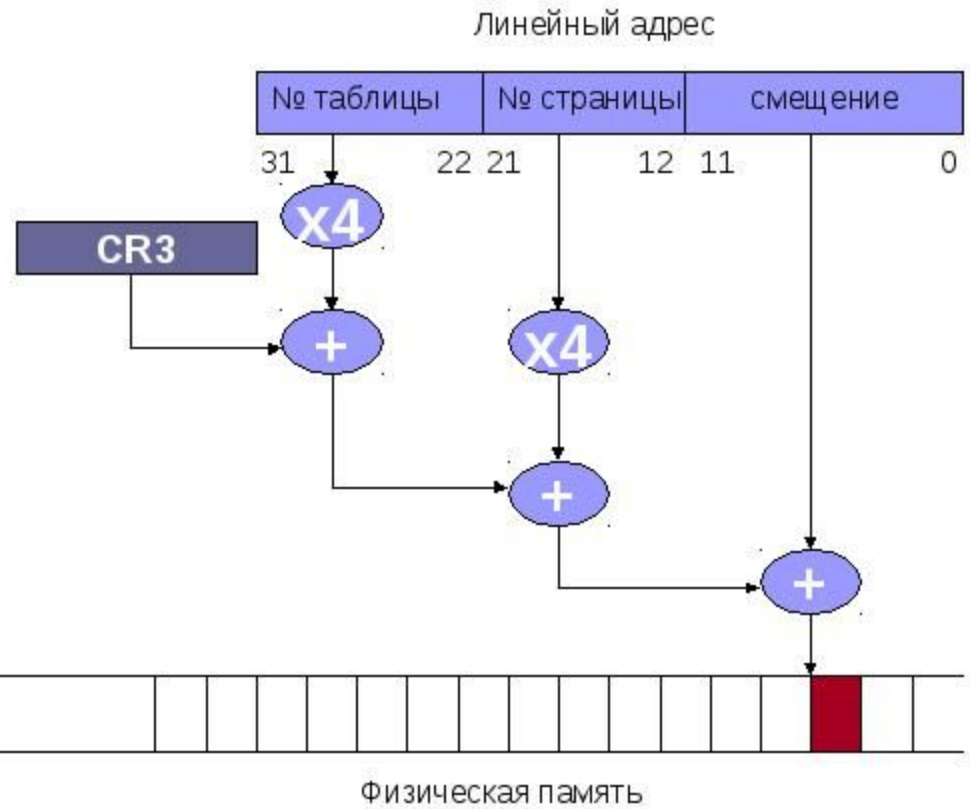
Страничная организация памяти: современный UNIX

- Глобальная таблица страниц хранит линейные адреса страниц в первичной и вторичной памяти
- Каждый процесс имеет частную таблицу страниц
- Глобальная и частные таблицы страниц используются для преобразования виртуальных адресов в физические
- Используется секционная организация программ
- Виртуальный адрес состоит из виртуального адреса начала области (секции) и смещения ячейки внутри секции



Страничная организация памяти в защищенном режиме процессора Intel и Microsoft Windows

- Страничная организация прозрачна для программиста – реализуется полностью аппаратно и на уровне ОС
- Рассматривается как надстройка над сегментной организацией памяти
- Используется для оптимизации управления памятью в ОС



Сегментно-страничная организация памяти

- Сегментная организация памяти удобна при организации программ
- Страничная организация памяти удобна при реализации свопинга
- Сегментно-страничная организация памяти реализуется как
  - Объединение сегментной и страничной организации – память делится на сегменты, а затем на страницы
  - Наслоение сегментной и страничной организации – в основе используется сегментная организация, но для окончательного преобразования виртуальных адресов - страничная (Intel)

Свопинг

- Свопинг – это механизм перемещения участков оперативной памяти между физической памятью и областью дискового пространства (областью свопинга или подкачки)
- Современные ОС реализуют пейджинг – частный случай свопинга, при котором обмен между ОП и областью свопинга выполняется участками памяти равными одной странице
- В ОС Windows этот механизм называется механизмом подкачки

Методы выполнения свопинга

- По требованию – участок памяти подгружается в ФОП в тот момент, когда процесс обращается к адресу памяти в этом участке
- Рабочие множества – ОС пытается предсказать, какие участки памяти понадобятся процессу в ближайшее время, учитывая принцип локальности

Типы распределения памяти

Управление основной памятью сводится в основном к распределению памяти. Возможно два типа распределения: статическое и динамическое.

Статический принцип уменьшает до минимума временные системные затраты, однако, не обеспечивает гибкости, что может привести к значительным издержкам памяти.

При динамическом подходе наоборот, больший эффект достигается за счет оптимального выделения памяти. Средства программирования на языках высокого уровня обеспечивают как статическое, так и динамическое распределение памяти. Правила для использования того или иного вида распределения определены в так называемой модели памяти.

Модели памяти

В модели определены правила для формирования структуры исполняемой программы и адресов.

Модель памяти устанавливает ограничения на размеры сегментов кода и данных.

Выбор той или иной модели осуществляется на стадии компиляции, т.е. перед выполнением компиляции необходимо передать компилятору опцию или флаг, в котором указывается тип выбранной модели памяти.

Окончательное формирование модели происходит на стадии компоновки, когда к пользовательскому коду добавляется системный код и данные.

В результате этих двух этапов создается загрузочный модуль, состоящий из сегментов с зарезервированными именами.

Загрузочный модуль состоит из двух секций: секции кода и секции данных.

Функции ОС по управлению основной памятью

Функции ОС по управлению основной памятью:

1. Учет свободных участков организуется связыванием всех свободных участков в цепочку. Каждый свободный участок представляется элементом, располагаемым в начале участка. В начале каждого участка располагается структура, которая содержит адрес следующего участка. Последний участок в поле адреса содержит ноль. Адрес первого элемента цепочки хранится в ОС.
2. Выделение участка памяти по запросам: элемент исключается из цепочки или изменяется его указатель длины.
3. Освобождение участка - наоборот.
4. Борьба с фрагментацией. Многократное выделение и освобождение памяти, разбиение пространства на множество участков. Последствием этого может стать невозможность выделения памяти по запросу программы.

Способы борьбы с фрагментацией:

- Расширение размера ОП (в том числе и за счет виртуализации).
- Упорядочение запросов на память. Запросы, подчиняющиеся какой-либо классификации, направляются в одни и те же области памяти.

5. Многоуровневое распределение основной памяти состоит в том, что выделенный участок на верхнем уровне подлежит дальнейшему распределению на нижнем. Существует три уровня распределения:

I. уровень заданий;
II. уровень задач;
III. уровень запросов.

Методы защиты памяти

Подсистема защиты памяти представляет собой комплекс аппаратно-программных средств, обеспечивающих предотвращение взаимного искажения одновременно находящихся в ОП программ и несанкционированного доступа к любой хранящейся в ОП информации.

В общем случае защита осуществляется как при записи для предотвращения искажения информации, не относящейся к выполняемой в данный момент программе, так и при считывании для исключения возможности использования информации, не принадлежащей данному пользователю, т.е. для предотвращения несанкционированного доступа к информации.

Независимо от принятых принципов построения подсистемы защиты памяти в основе её функционирования заложена проверка всех адресов, поступающих для обращения к ОП.

В результате такой проверки формируется сигналы управления, разрешающий обращение к ОП, если адрес относится к выделенной для данной программы области памяти, в противном случае вы рабатывается сигнал, запрещающий выполнение данной команды (при этом посылается запрос на прерывание реализуемой программы с целью установления причины нарушения границ разрешенной для использования области памяти).

Реализация идеи защиты памяти в любом случае не должна сопровождаться заметным снижением производительности машины и не требовать больших аппаратных затрат.

Различают три способа защиты памяти:

- по граничным адресам,
- по маскам
- по уровням привилегий (ключам).

Защита памяти по граничным адресам

Защита памяти по граничным адресам осуществляется с помощью регистров и узлов сравнения кодов, размещаемых в блоке защиты памяти (БЗП).
Реализация этого способа защиты предусматривает выделение для каждой программы определенной области ОП, составленной из ячеек с последовательными адресами.
Границы области отмечаются фиксированием адресов её начальной и конечной ячеек.
Граничные адреса вводятся в регистры БЗП управляющей программой операционной системы перед началом выполнения каждой рабочей программы. При выполнении данной рабочей программы каждый поступающий в ОП исполнительный адрес с помощью узлов сравнения кодов сравнивается с граничными адресами.
По результатам сравнения устанавливается возможность обращения к ОП по поступившему адресу: если он находится в пределах граничных адресов, то разрешается доступ к соответствующей ячейке памяти, в противном случае вырабатывается сигнал запроса на прерывание выполняемой программы.

Преимущество данного способа защиты памяти состоит в том, что он позволяет защищать области памяти произвольной длины.
Кроме того, блок защиты достаточно прост, а его функционирование не приводит к значительным временным затратам.
Однако необходимость размещения программ в областях памяти с последовательными номерами ячеек и ограниченных двумя граничными адресами существенно снижает возможности программирования и даже эффективность работы ЭВМ.
Поэтому способ защиты памяти по граничным адресам в настоящее время применяется редко, при статическом распределении памяти, когда для каждой из параллельно выполняемых рабочих программ заранее (до начала их выполнения) отводится определенная область памяти.

Защита памяти по маскам

Защита памяти по маскам используется при страничной организации ОП.

Для каждой программы перед её выполнением указываются номера страниц, отведенные для размещения её команд и всех необходимых данных.

Указание о номерах отведенных страниц для данной программы задается управляющей программой операционной системы в виде кода маски или кода признаков страниц.

Код маски формируется для каждой рабочей программы.

Под маской программы понимается n-разрядный двоичный код, разрядность которого определяется количеством страниц ОП.

Каждый i-й разряд маски указывает о принадлежности i-й страницы ОП данной программе: если в i-м разряде задано значение 1, то при обращении к ОП разрешен доступ к любой ячейке i-ой страницы, если же i-й разряд маски содержит ноль, то выполняемой программе доступ к i-й странице запрещен.

По сравнению с защитой по граничным адресам защита памяти по маскам отличается большей гибкостью при организации распределения ОП.

Для своей реализации данный метод не требует сложного оборудования.

Однако при большой емкости ОП, состоящей из значительного количества страниц, она становится неэффективной. Это связано с многоразрядно-стью кода маски, возрастанием сложности дешифратора номера страниц и всего БЗП, а также заметным увеличением времени работы БЗП по формированию управляющих сигналов.

Защита памяти по ключам

Защита памяти по ключам (уровням привилегий) используется в большинстве современных многопрограммных ЭВМ со страничной организацией памяти и динамическим её распределением между параллельно вы полняемыми программами.

В её основе лежит применение специальных кодов (уровней) для проверки соответствия используемых массивов ячеек памяти номеру выполняемой программы.

Каждой рабочей программе ОС придает специальный ключ — ключ программы.

Все выделенные для данной рабочей программы страницы отмечаются одним и тем же ключом страницы или ключом защиты.

В качестве ключа защиты обычно указывается двоичный код номера программы.

В процессе обращения к ОП производится сравнение ключа выполняемой про граммы с ключами защиты соответствующих страниц памяти.

Обращение разрешается только при совпадении сравниваемых кодов ключей. Защита памяти по ключам применяется не только при работе ОП с процессором, но и в ходе обмена информацией с ВЗУ через каналы ввода-вывода.

Тогда вместо ключей программ используются ключи каналов. Разрядность кодов ключей определяется максимальным количеством параллельно выполняемых программ.

Защита памяти по ключам (уровням привилегий) является наиболее универсальной и гибкой, особенно эффективной при страничной или сегментно-страничной организации виртуальной памяти и динамическом её распределении.

Однако такой способ защиты требует для своей технической реализации заметных дополнительных аппаратных затрат, прежде всего необходима ПКЗ с очень небольшим временем выборки кода ключей. Память ключей защиты строится так, чтобы время выборки кода было практически на порядок меньше времени выборки из ОП.