Основы управления памятью

Методы распределения памяти с использованием дискового пространства

Методы распределения памяти с использованием дискового пространства

Виртуальная память

Оверлеи

- Уже достаточно давно пользователи столкнулись с проблемой размещения в памяти программ, размер которых превышал имеющуюся в наличии свободную память.
- Одним из решений этой проблемы было разбиение программы на части, называемые *оверлеями*.
- 0-ой оверлей начинал выполняться первым. Когда он заканчивал свое выполнение, он вызывал другой оверлей.
- Все оверлеи хранились на диске и перемещались между памятью и диском средствами ОС.

Достоинства и недостатки оверлеев

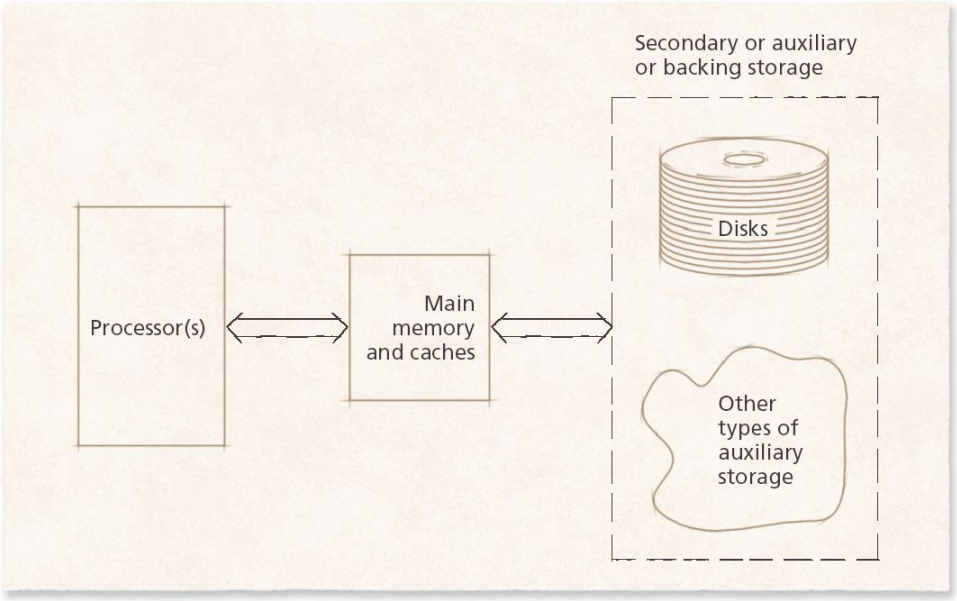
- Достоинства
  - Возможность запускать программы размером большим, чем размер оперативной памяти;
  - Не требуется специальная поддержка со стороны операционной системы.
- Недостатки
  - Программист должен самостоятельно разработать и реализовать структуру оверлеев.
  - Подобная разработка больших программ может вызвать затруднения в реализации , поэтому автоматическая реализация (поддержка со стороны операционной системы) в этом случае будет предпочтительнее.

История возникновения виртуальной памяти

- Развитие методов организации вычислительного процесса в этом направлении привело к появлению метода, известного под названием *виртуальная память*.
- Термин *виртуальная память* обычно ассоциируется с возможностью адресовать пространство памяти, гораздо большее, чем емкость основной (оперативной) памяти конкретной вычислительной машины, благодаря использованию хранилища на вторичной памяти (например, на жестком диске).
- Впервые концепция виртуальной памяти была реализована в вычислительной машине Atlas, созданной в Манчестерском университете в Англии в 1962г.

Понятие «виртуальной памяти»

- Виртуальная память (ВП) – это совокупность программно-аппаратных средств, позволяющих выполнять программы, размер которых превосходит имеющуюся оперативную память.

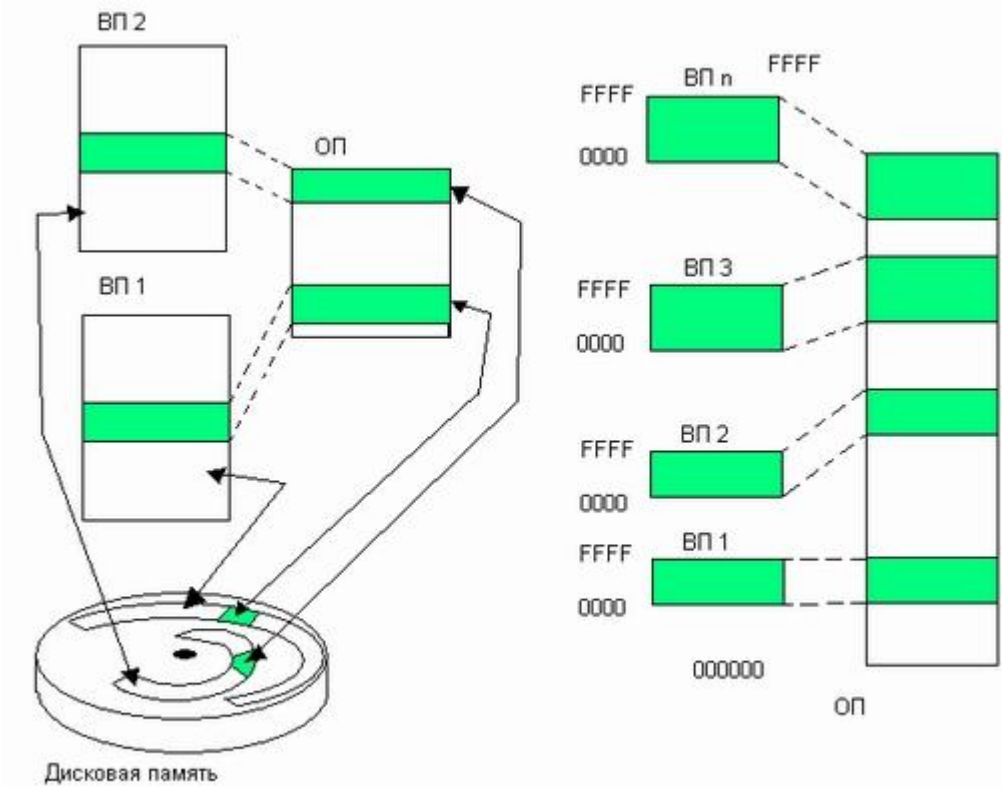


Менеджер виртуальной памяти

- Реализацией механизма виртуальной памяти занимается специальный менеджер.
- Менеджер виртуальной памяти выполняет три основные задачи:
  - размещение данных в запоминающих устройствах разного вида;
  - перемещение данных между запоминающих устройств разного вида;
  - преобразование адресов.

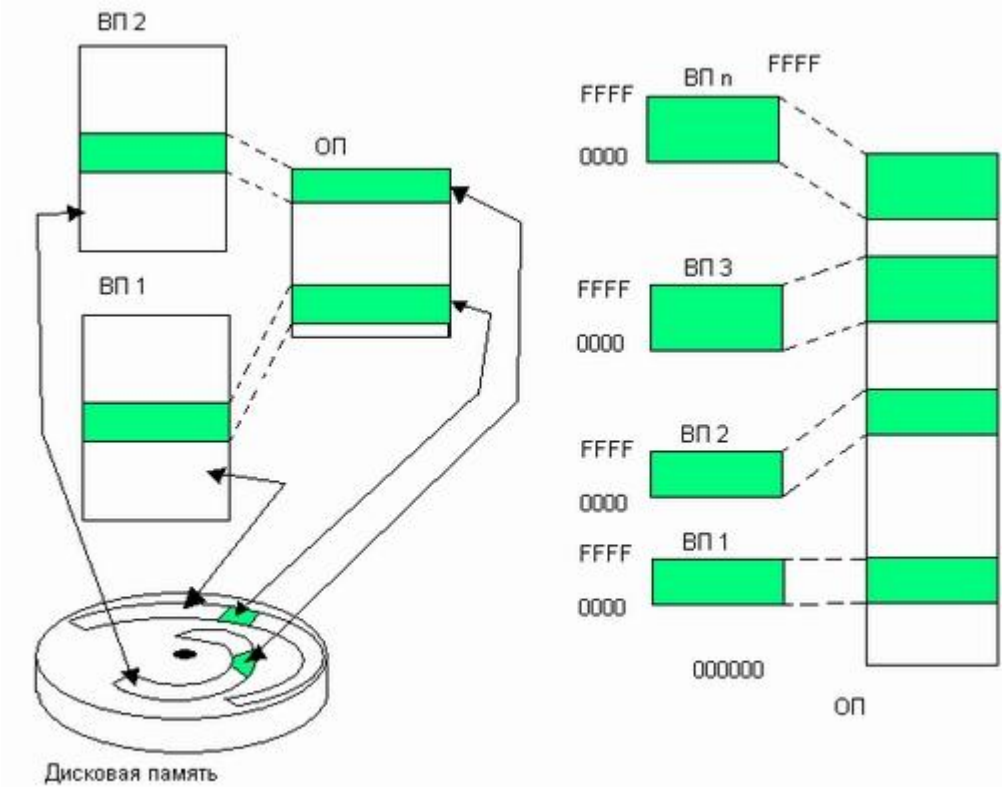
Задачи менеджера ВП (1)

1. Размещает данные в запоминающих устройствах разного типа, например, часть программы в оперативной памяти, а часть на диске.



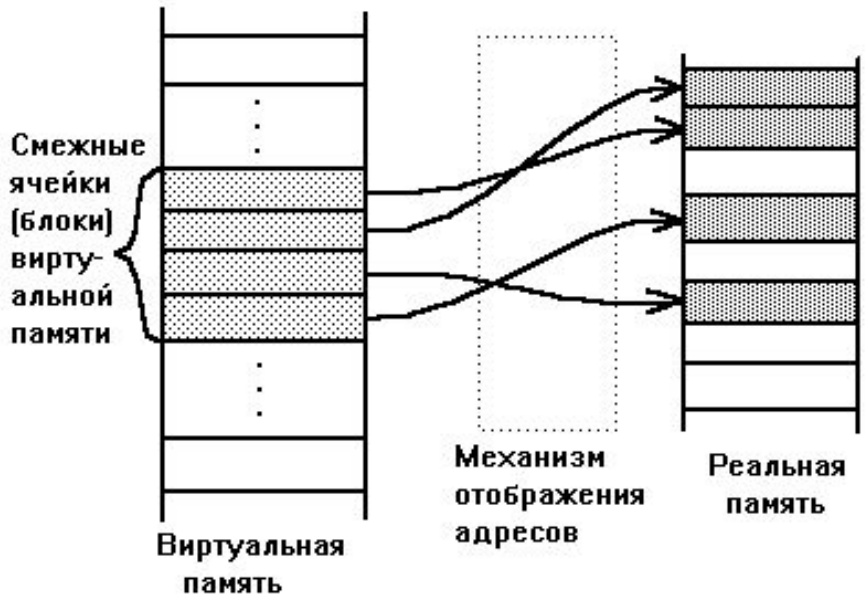
Задачи менеджера ВП (2)

2. Перемещает по мере необходимости данные между запоминающими устройствами разного типа, например, подгружает нужную часть программы с диска в оперативную память (свопинг).



Задачи менеджера ВП (3)

3. Преобразует виртуальные адреса (т.е. адреса, которыми оперирует процесс) в физические (т.е. адреса, которые реально существуют в оперативной памяти).



Обратите внимание, что смежные адреса виртуального адресного пространства процесса не обязательно будут смежными в реальной памяти, это свойство называют «искусственной смежностью».

Способы организации виртуальной памяти

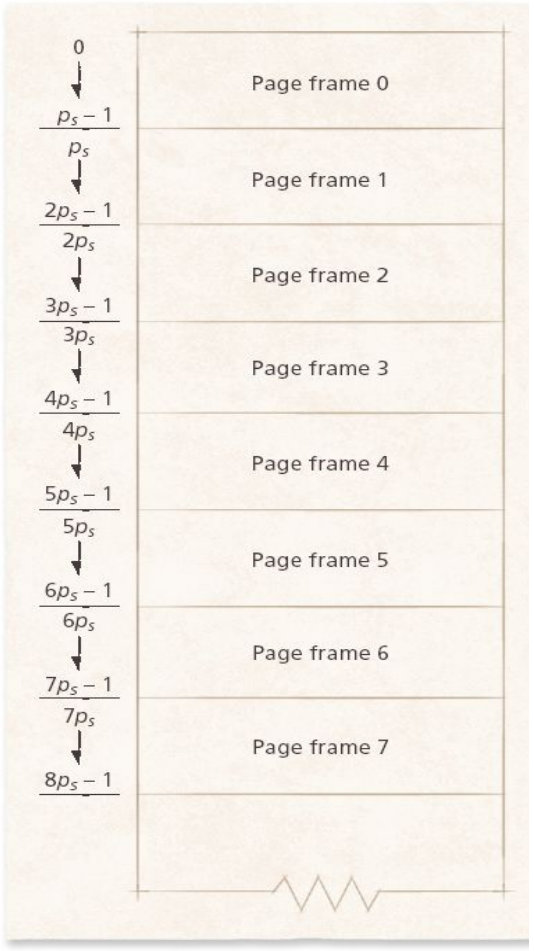
- страничное распределение
- сегментное распределение
- сегментно-страничное распределение

Методы распределения памяти с использованием дискового пространства

Страничное распределение

Разбиение памяти на страницы

- Оперативная память и виртуальное адресное пространство (ВАП) каждого процесса делится на фрагменты одинакового, фиксированного для данной ОС размера, называемые страницами.
- Размер страницы кратен степени двойки, это позволяет упростить механизм преобразования адресов.



Формат виртуального адреса

- При таком разбиение ВАП на страницы виртуальный адрес будет иметь в общем случае следующий вид:

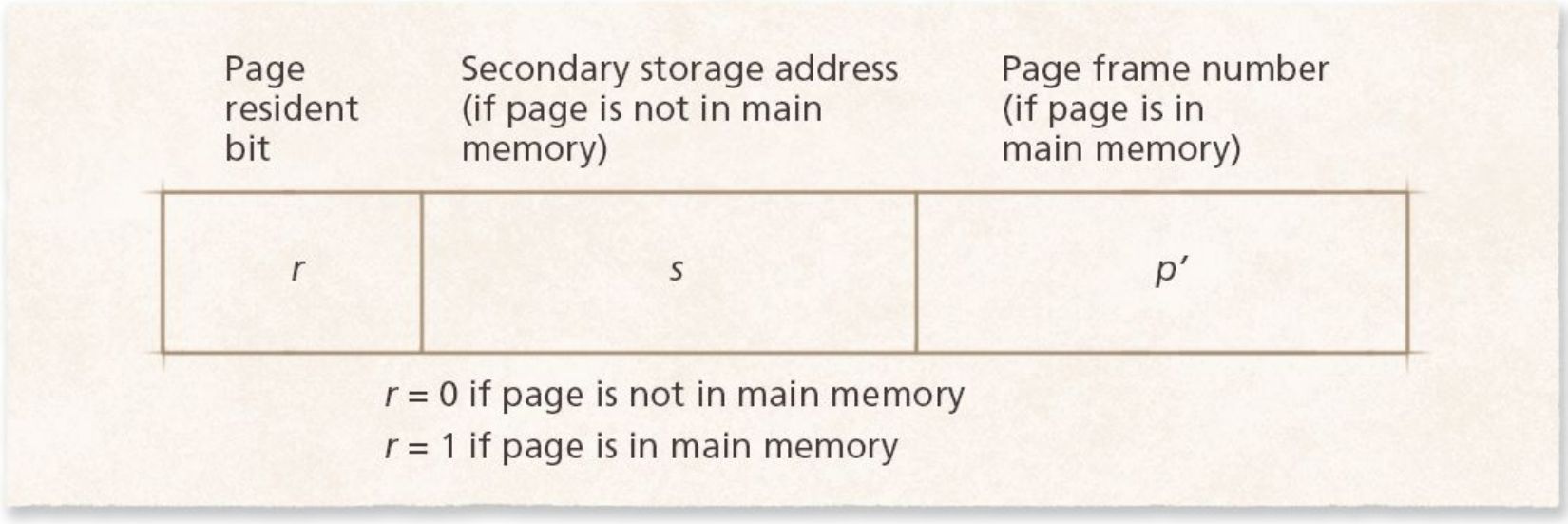
- где *p* – номер виртуальной страницы процесса (нумерация страниц начинается с 0), а *d* – смещение адресуемого байта в пределах страницы.

Таблица страниц процесса

- Система отображения виртуальных адресов в физические сводится к системе отображения виртуальных страниц в физические и представляет собой таблицу страниц.
- Для каждого процесса операционная система создает для каждого процесса одну или несколько таблиц страниц, которые устанавливают соответствие между номерами виртуальных и физических страниц для страниц, загруженных в ОП, или делается отметка о том, что виртуальная страница выгружена на диск.
- Для ссылки на таблицу страниц используется специальный регистр процессора. При распределении очередного процесса на выполнение в этот регистр загружается адрес таблицы страниц нового процесса.

Элемент таблицы страниц

- Устанавливает соответствие виртуального адреса страницы *p* физическому адресу страницы *p′*.
- Бит присутствия *r* указывает на загрузку страницы в оперативную память.

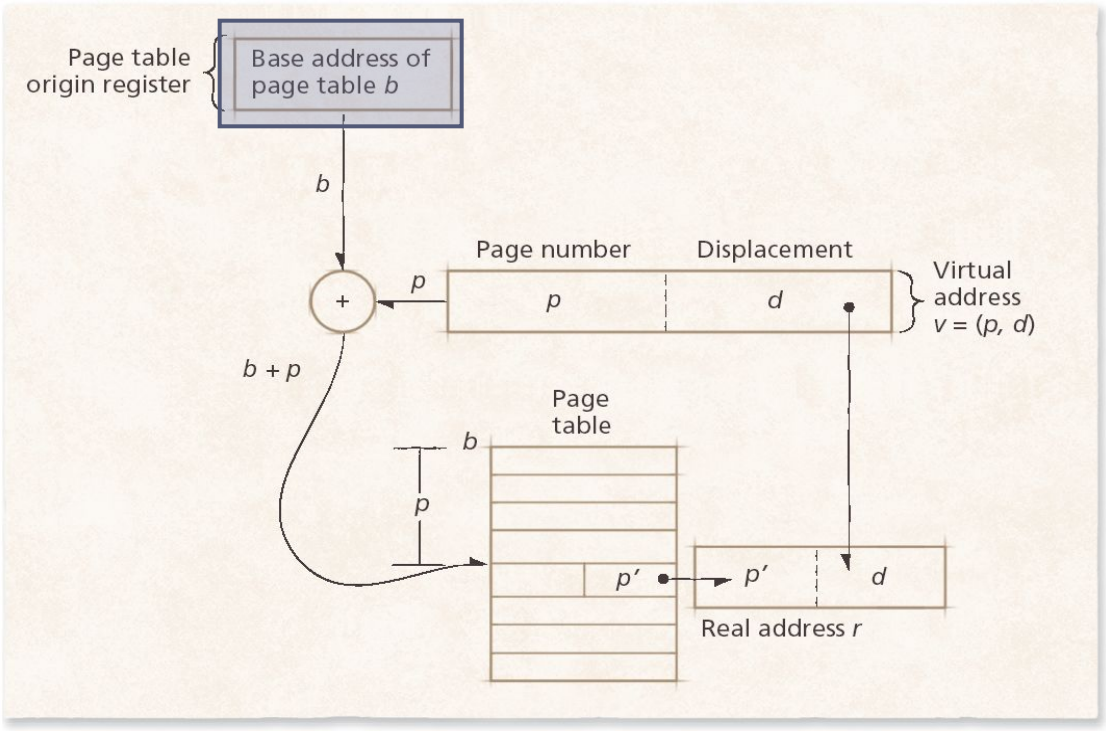


Перемещение страницы из вторичной памяти в первичную

- При каждом обращении к памяти происходит чтение из таблицы страниц информации о виртуальной странице, к которой произошло обращение.
  - Если данная виртуальная страница находится в ОП, то выполняется преобразование ВА в ФА.
  - Если же нужная виртуальная страница в данный момент выгружена на диск, то происходит так называемое страничное прерывание.
    - Программа обработки страничного прерывания находит на диске требуемую виртуальную страницу и пытается загрузить ее в ОП.
    - Если в памяти имеется свободная физическая страница, то загрузка выполняется немедленно, если же свободных страниц нет, то решается вопрос, какую страницу следует выгрузить из ОП (свопинг).

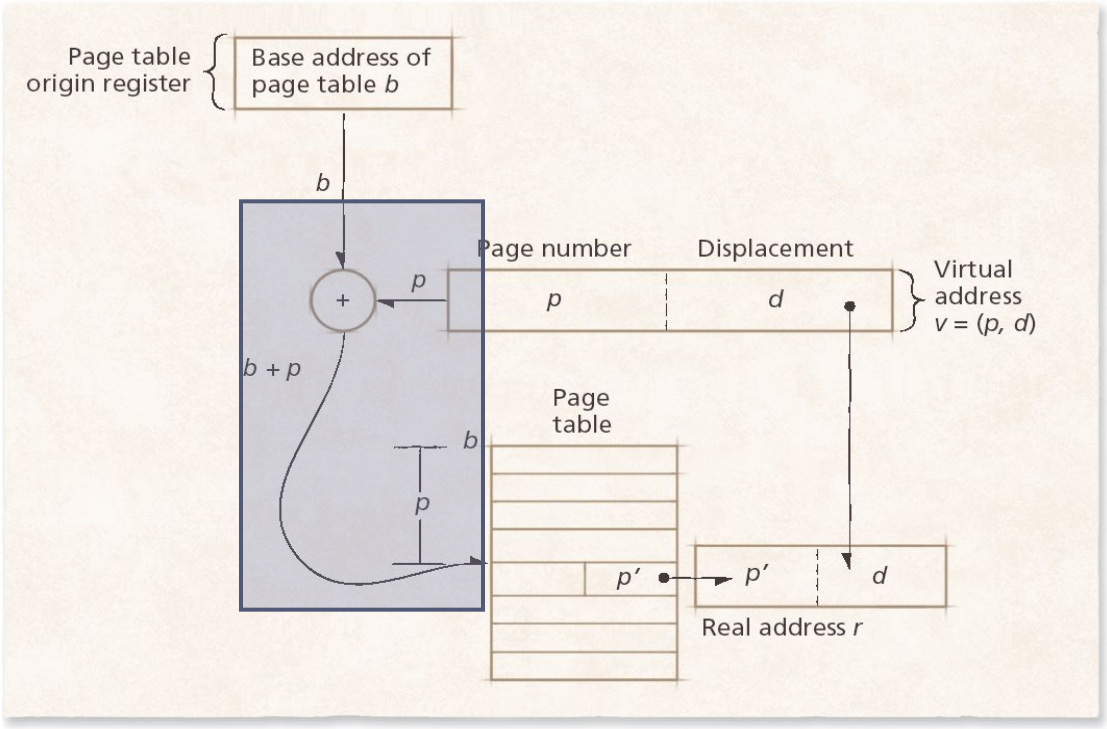
Трансляции адреса в страничной системе (1)

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



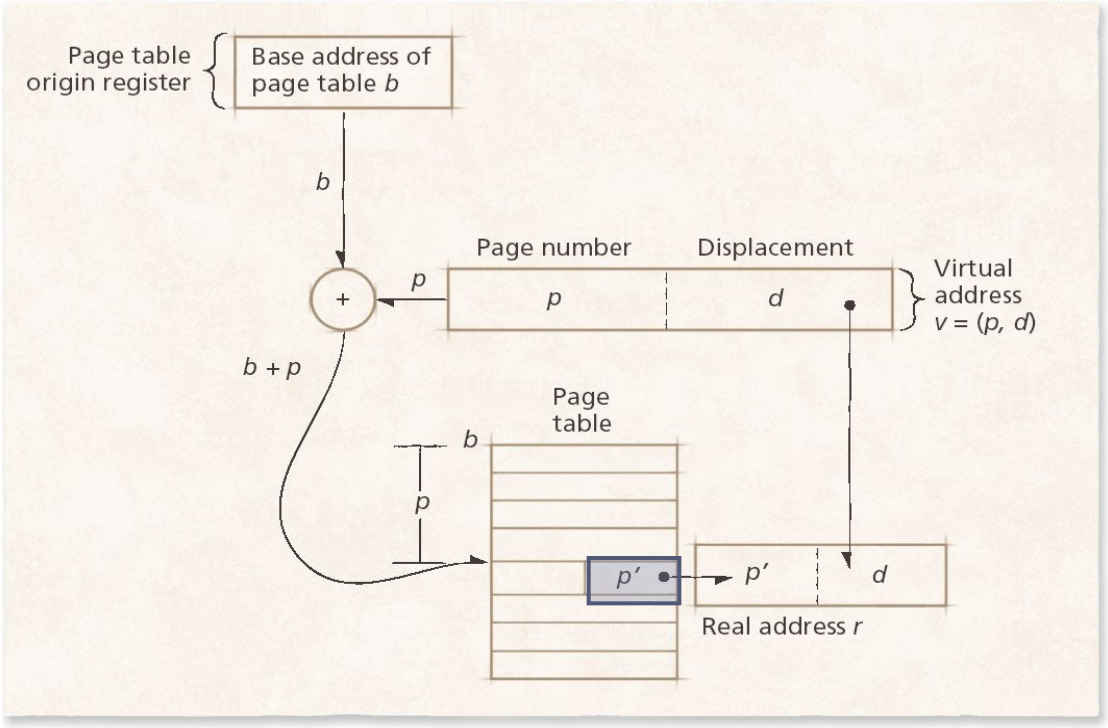
Трансляции адреса в страничной системе (2)

2. На основании начального адреса таблицы страниц, номера виртуальной страницы и длины записи в таблице страниц определяется адрес нужной записи.



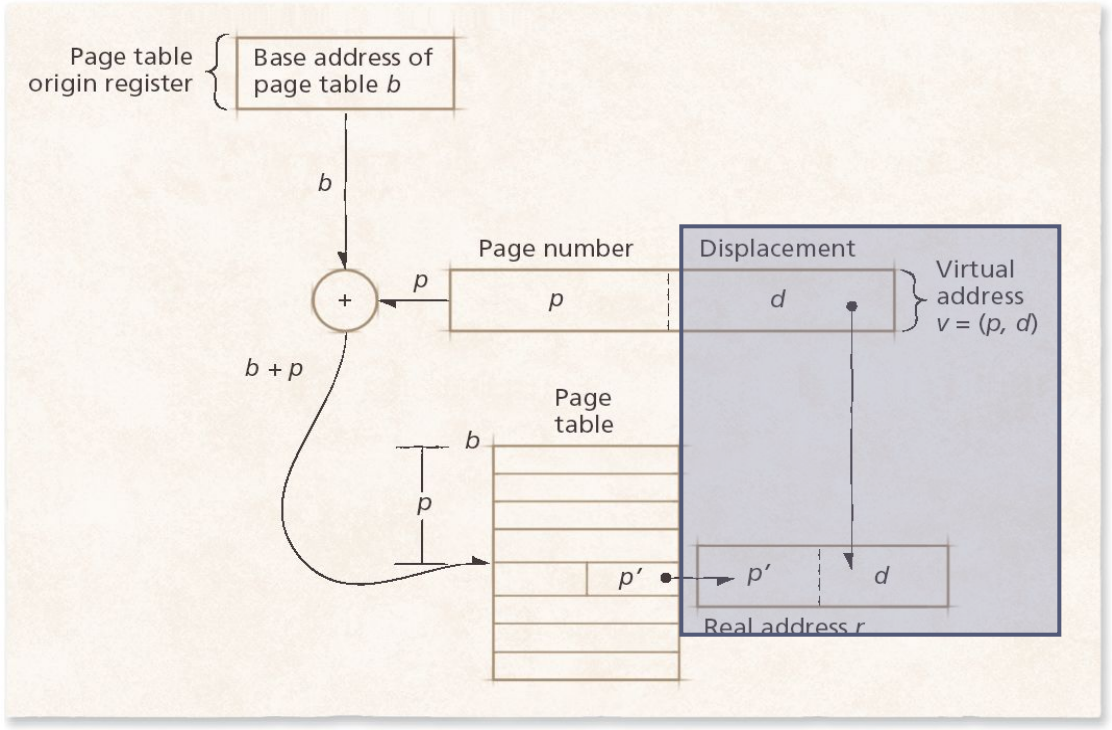
Трансляции адреса в страничной системе (3)

3. Из этой записи таблицы страниц извлекается номер физической страницы.

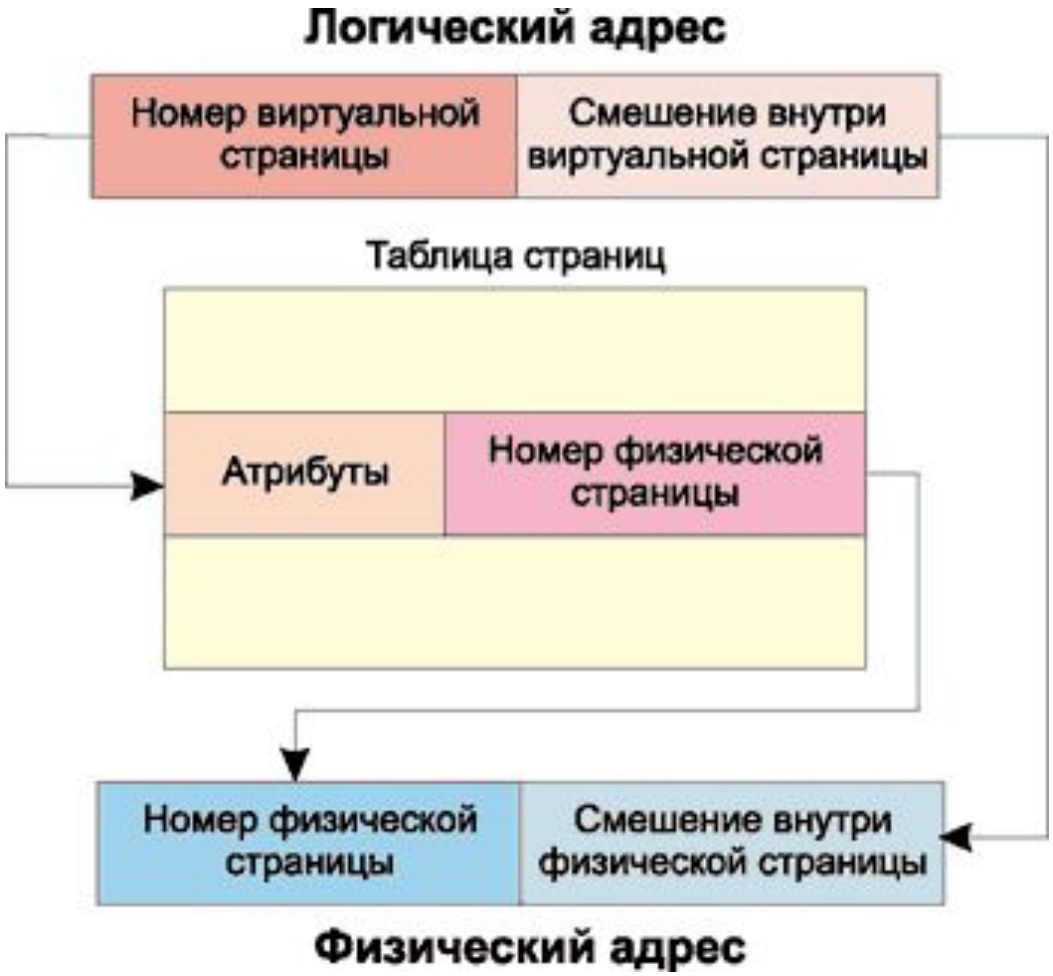


Трансляции адреса в страничной системе (4)

4. Учитывая, что размер страницы кратен 2, то младшие разряды смещения виртуального адреса могут быть просто присоединены номеру физической страницы.

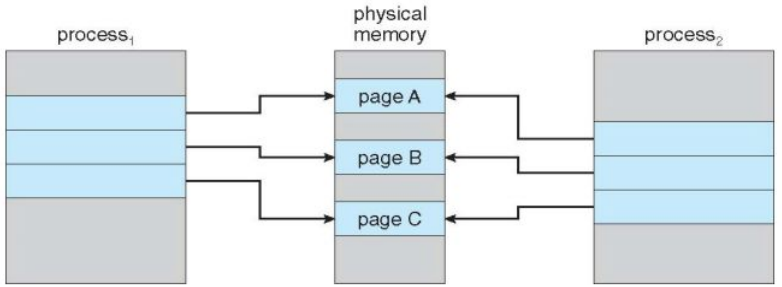


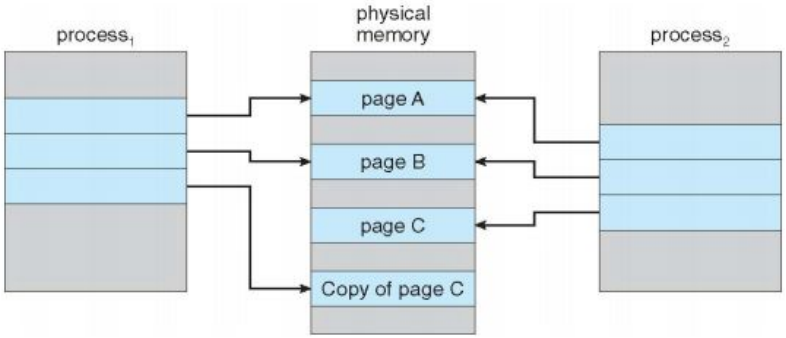
Страничное распределение: преобразование ВА в ФА



Технология Copy-on-Write

- Технология Copy-on-Write (COW) реализует совместное использование одних и тех же страниц физической памяти несколькими процессами.
- Технология разрешает процессам совместное использование страниц в режиме чтения.
- Как только один из процессов выполняет запись в общую память, то для этого процесса создается своя копия страницы.





Применение Copy-on-Write

- Технология Copy-on-Write применяется в UNIX-системах для экономии памяти и ускорения запуска дочерних процессов.
- Когда один процесс (родительский) порождает дочерний процесс (с помощью функции *fork()*), то дочерний процесс получает доступ к физическим страницам памяти родительского процесса, после чего обоим процессам запрещается запись в эти страницы.
- При попытке записи в разделяемые страницы сработает механизм защиты, он вызовет функцию обработки исключения, в результате выполнения которой измененная страница будет скопирована для монопольного доступа изменившего ее процесса.

Достоинства и недостатки страничного распределения

- Преобразование виртуального адреса в физический использует быструю операцию конкатенации (присоединения), что уменьшает время получения ФА.
- Отсутствует фрагментация виртуального адресного пространства.
- Большой размер таблиц страниц.
- Возможное неэффективное использование физических страниц памяти.
- Потенциально большое количество страничных прерываний (смежные данные могут быть на разных страницах).
- Сложность реализации защиты данных (каждой странице необходимо устанавливать атрибуты защиты).

Выбор размера страницы

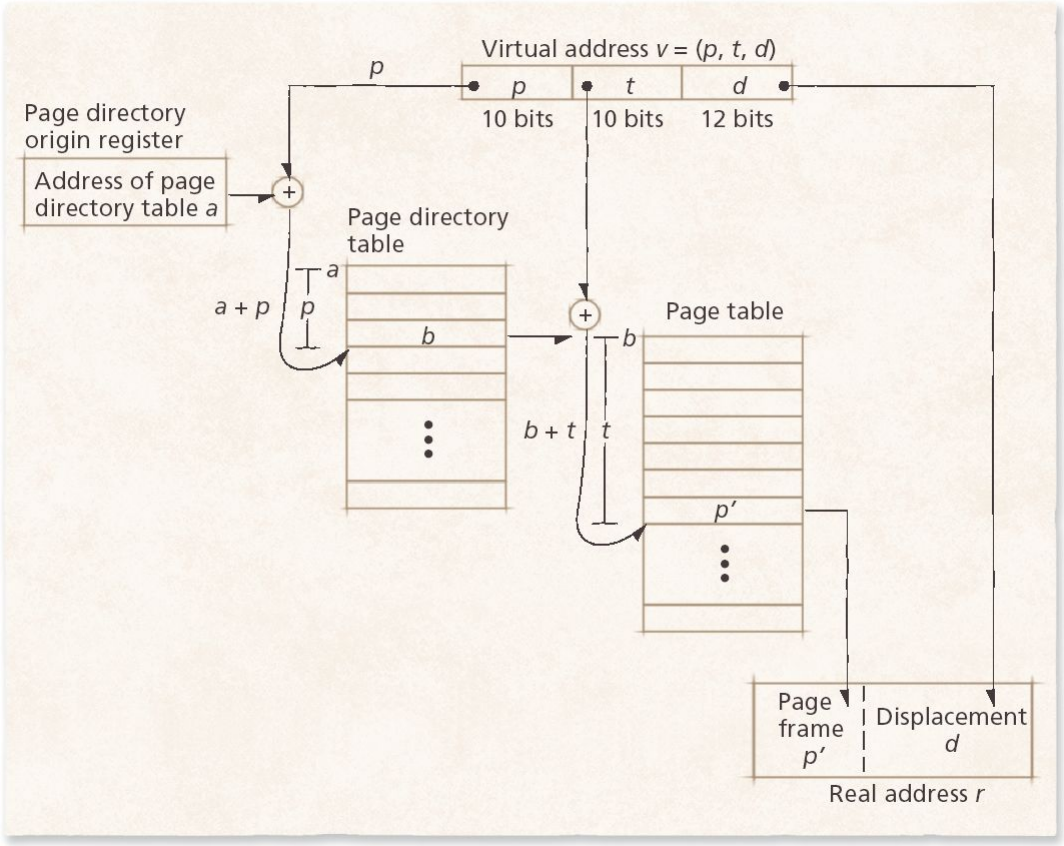
- при малых страницах:
  - меньшая внутренняя фрагментация страниц и повышается эффективность использования оперативной памяти;
  - снижаются расходы времени на свопинг страниц (т.к. в памяти помещается больше страниц);
- при больших страницах:
  - меньшие затраты на поиск и управление страницами (таблицы имеют меньший размер);
  - выше эффективность обмена с внешней памятью.

Расчет размера таблицы страниц

- Характеристика вычислительной системы:
  - 32 битное виртуальное адресное пространство
  - размер страницы – 4Кбайта
  - 4 Гигабайта физической памяти
- Количество элементов таблицы страниц:
  - 2^{32} адресуемых байт / 2^{12} байт на страницу = 2^{20} записей
- Размер элемента таблицы страниц:
  - 4 Гигабайта физической памяти = 2^{32} байт
    - 2^{32} байт памяти / 2^{12} байт на страницу = 2^{20} физических страниц
    - 20 бит необходимо для хранения физического номера страницы
  - Элемент таблицы страниц = ~4 байт
    - флаги доступа + номер физической страницы
    - 20 бит для хранения физического номера страницы = ~3 байта
    - флаги доступа = ~1 байт
- Размер таблицы страниц на 1 процесс (!):
  - 2^{20} записей * 2^{2} байт = 2^{22} байт (4 Мегабайта)

Использование многоуровневых таблиц страниц

- Для того чтобы избежать необходимости постоянного хранения в памяти огромных таблиц, применяются многоуровневые таблицы страниц.
- Например, для двухуровневой таблицы страниц обычно достаточно хранить таблицу верхнего уровня и несколько таблиц второго уровня.



Примеры реализации многоуровневых таблиц страниц

- Количество уровней в *таблице страниц* зависит от конкретных особенностей архитектуры.
- Можно привести примеры реализации одноуровневого (DEC PDP-11), двухуровневого (Intel, DEC VAX), трехуровневого (Sun SPARC, DEC Alpha) *пейджинга*, а также *пейджинга* с заданным количеством уровней (Motorola).
- Функционирование RISC-процессора MIPS R2000 осуществляется вообще без *таблицы страниц* (так называемый zero level paging).

Вопрос

- Какой будет формат физического адреса при использовании трехуровневой таблицы страниц?

Методы распределения памяти с использованием дискового пространства

Ускорение преобразования страничных адресов

Ускорение преобразования страничных адресов

- При каждом обращении к памяти выполняется табличное преобразование виртуального адреса в физический и как следствие:
  - доступ к таблице страниц для считывания физического адреса страницы;
  - доступ к физической памяти.
- Таким образом, мы получаем при каждом обращении по виртуальному адресу **ДВА обращения** к физической памяти.
- Если используются многоуровневые таблицы страниц, то количество обращений возрастает пропорционально глубине таблицы страниц.

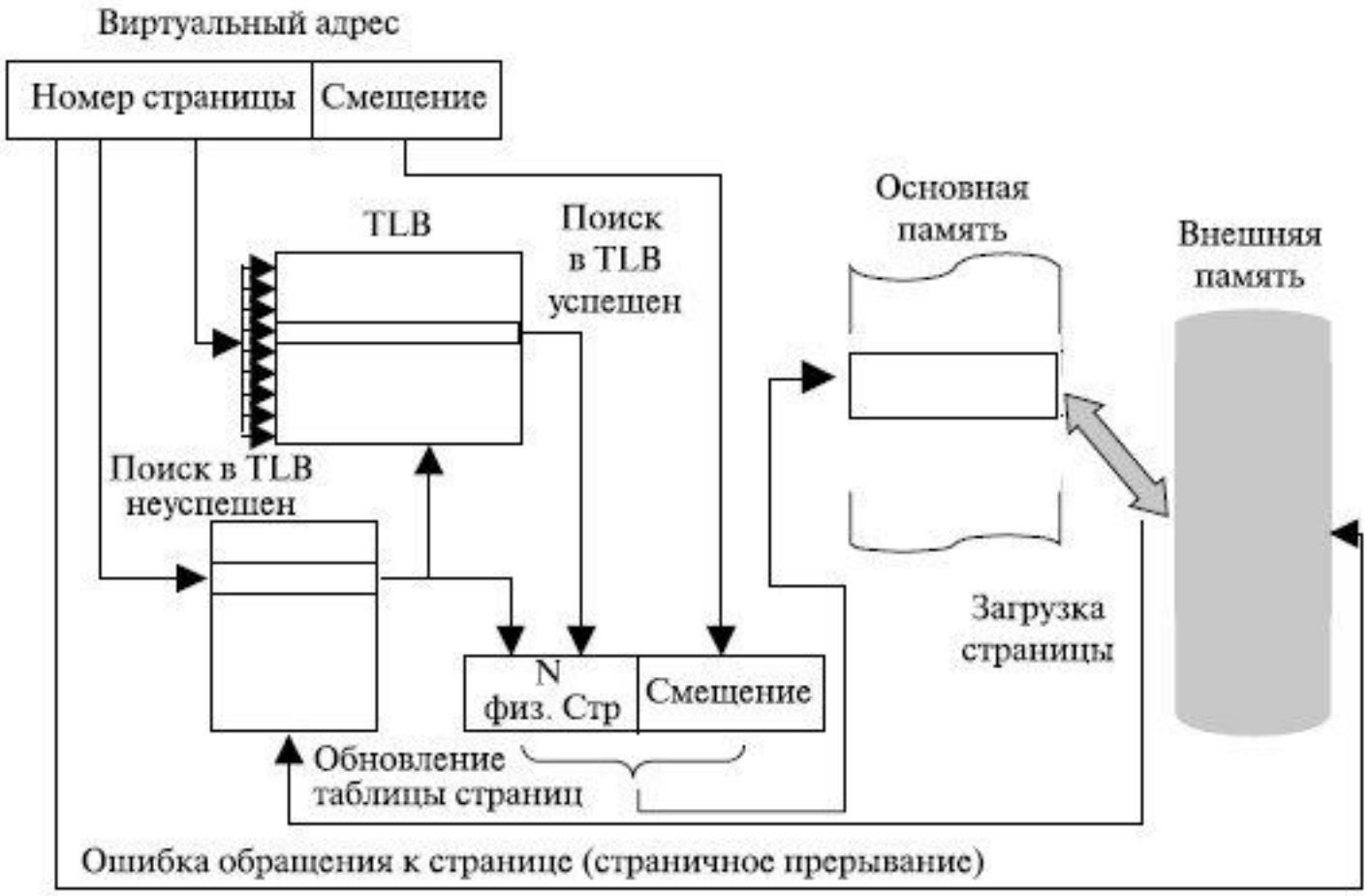
Ускорение преобразования страничных адресов

Ассоциативный буфер трансляции

Ассоциативный буфер трансляции

- Для ускорения процесса преобразования адреса современные процессоры используют ассоциативный буфер трансляции (translation look-aside buffer, TLB), для кэширования наиболее частых связей между физической и виртуальной памятью.
- Кэш TLB представляет собой вектор, ячейки которого можно считывать и сразу сравнивать с целевым значением. В случае TLB вектор содержит сопоставления физических и виртуальных адресов для недавно использовавшихся страниц, а также атрибуты защиты каждой страницы
- Если запрашиваемый адрес не находится в кэше, то процессор, перед ответом на запрос, переходит к таблице страниц для поиска соответствия виртуального адреса физическому.
- TLB должен также предусматривать способы организации записей в кэш и принятия решения о том, какая из старых записей должна быть удалена при внесении в кэш новой записи.

Схема реализации TLB



Пример использования TLB

- Процессор аппаратно способен одновременно опрашивать все записи TLB для определения того, какая из них соответствует заданному номеру страницы.
- Такой подход известен как ассоциативное отображение (associative mapping).
- Обычно число записей в TLB составляет от 8 до 2048.

Повышение эффективности TLB

- Результативность поиска в TLB определенного размера с ростом размера ВАП процессов и уменьшением локализации снижается. В этом случае TLB может стать узким местом, ограничивающим производительность виртуальной памяти.
- Способы повышения производительности TLB:
  - использование большого TLB с большим количеством записей (аппаратное увеличение размера TLB);
  - использование страниц большего размера, с тем чтобы каждая запись в TLB ссылалась на большой блок памяти, однако использование больших размеров страниц может привести к потере производительности;
  - использование страниц разных размеров, что обеспечит необходимую для эффективного использования гибкость.

Проблемы использования TLB

- При переключении процессов нужно добиться того, чтобы новый процесс не видел в ассоциативной памяти информацию, относящуюся к предыдущему процессу, например, выполнять ее очистку.
- TLB-кэши многопроцессорной системы аппаратно не синхронизируются , ядро операционной системы должно само выполнять действия по синхронизации их содержимого.

Zero level paging

- Таблица страниц отсутствует вообще.
- TLB-буфер очень большой, чтобы хранить все (или большинство) преобразований виртуальных адресов в физические.
- Отсутствие преобразования адреса в ассоциативной памяти приводит к системному прерыванию (аналогичному с страничному прерыванию для загрузки отсутствующей страницы – *page fault*).
- В случае возникновения системного прерывания ОС должна выполнить поиск страницы, отсутствующей в TLB.

Ускорение преобразования страничных адресов

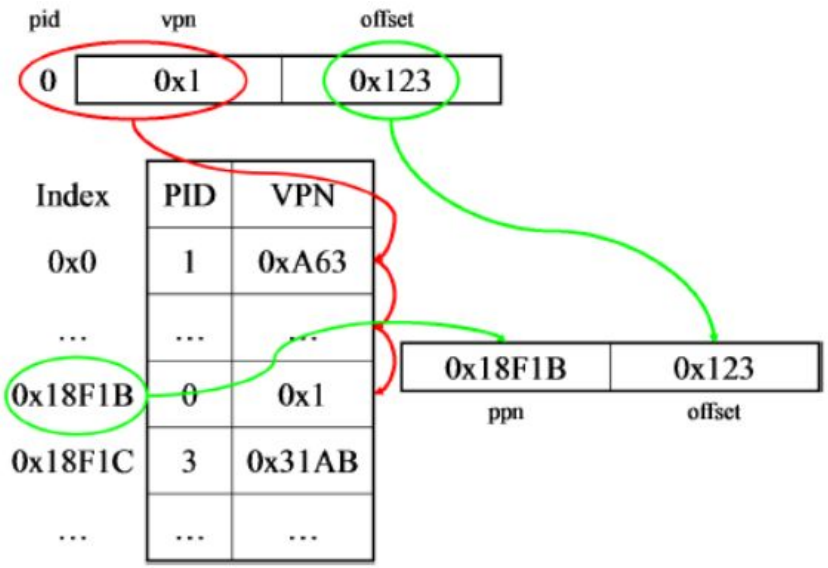
Инвертированные таблицы страниц

Инвертированные таблицы страниц

- Несмотря на многоуровневую организацию, хранение нескольких таблиц страниц большого размера по-прежнему представляют собой проблему. Ее значение особенно актуально для 64-разрядных архитектур, где число виртуальных страниц очень велико. Вариантом решения является применение инвертированной таблицы страниц (inverted page table). Этот подход применяется на машинах PowerPC, некоторых рабочих станциях Hewlett-Packard, IBM RT, IBM AS/400 и ряде других.
- В этой таблице содержится по одной записи на каждый страничный кадр физической памяти. В этом случае для всех процессов достаточно всего одной таблицы, причем размер этой таблицы не зависит разрядности архитектуры, размера и количества процессов.

Линейная инвертированная таблица страниц

- Элемент таблицы страниц содержит:
  - идентификатор процесса-владельца страницы (таблица является общей для всех процессов);
  - номер виртуальной страницы.
- Таблица проиндексирована по номеру физической страницы.

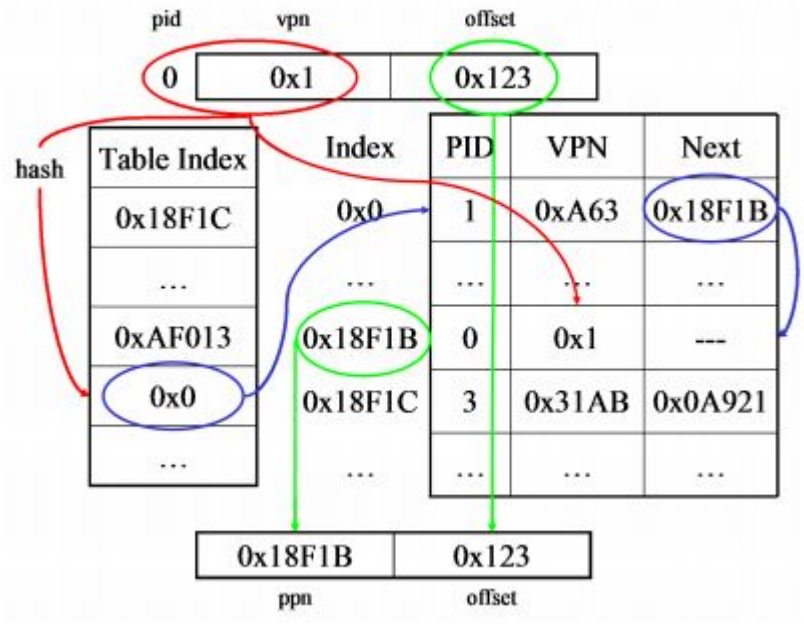


Проблемы использования инвертированных таблиц страниц

- Несмотря на экономию оперативной памяти, применение линейных инвертированных таблиц имеет существенный минус – записи в ней не отсортированы по возрастанию номеров виртуальных страниц, что приводит к необходимости выполнения большого количества обращения к памяти и большого количества операций сравнения, что сильно усложняет трансляцию адреса.
- Один из способов решения данной проблемы – использование хеш-таблицы виртуальных адресов. При этом часть виртуального адреса, представляющая собой номер страницы, отображается в хеш-таблицу с использованием функции хеширования.

Хешированная инвертированная таблица страниц

- Каждой странице физической памяти здесь соответствует одна запись в хеш-таблице инвертированной таблицы страниц.
- Виртуальные адреса, имеющие одно значение хеш-функции, сцепляются друг с другом. Обычно длина цепочки не превышает двух записей.



Вопрос

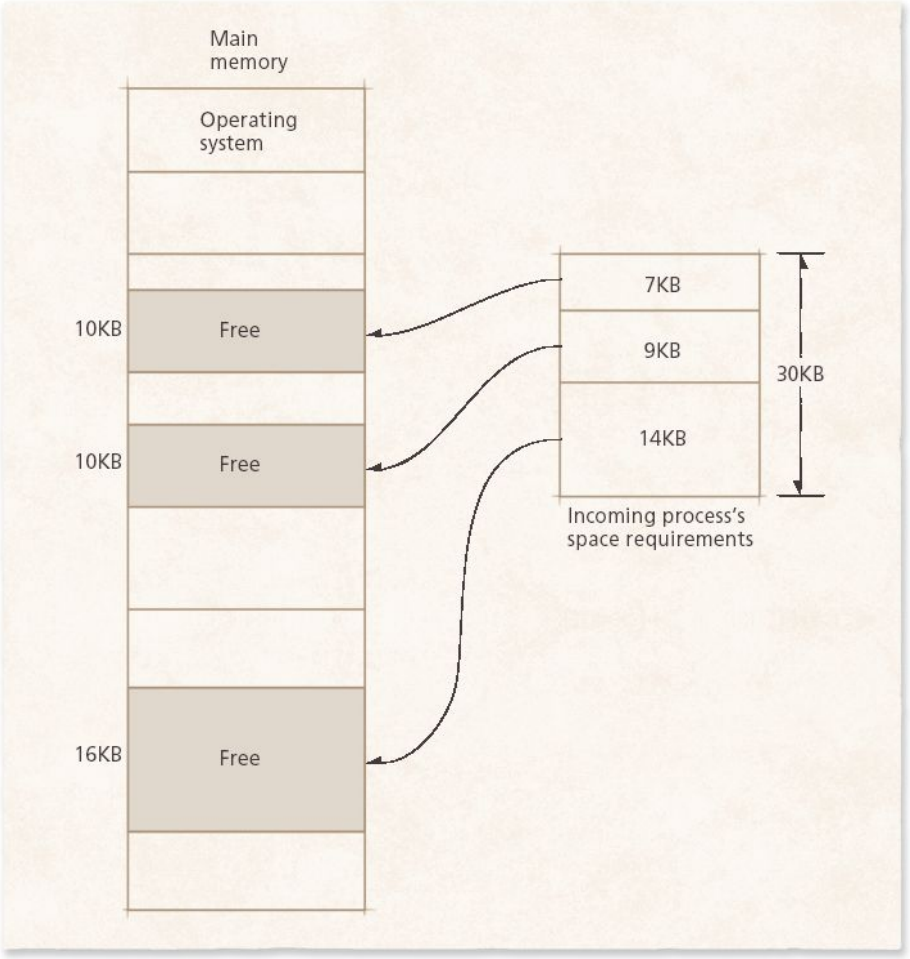
- От чего зависит размер инвертированной таблицы страниц?

Методы распределения памяти с использованием дискового пространства

Сегментное распределение

Сегментное распределение

- ВАП 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.



Формат виртуального адреса

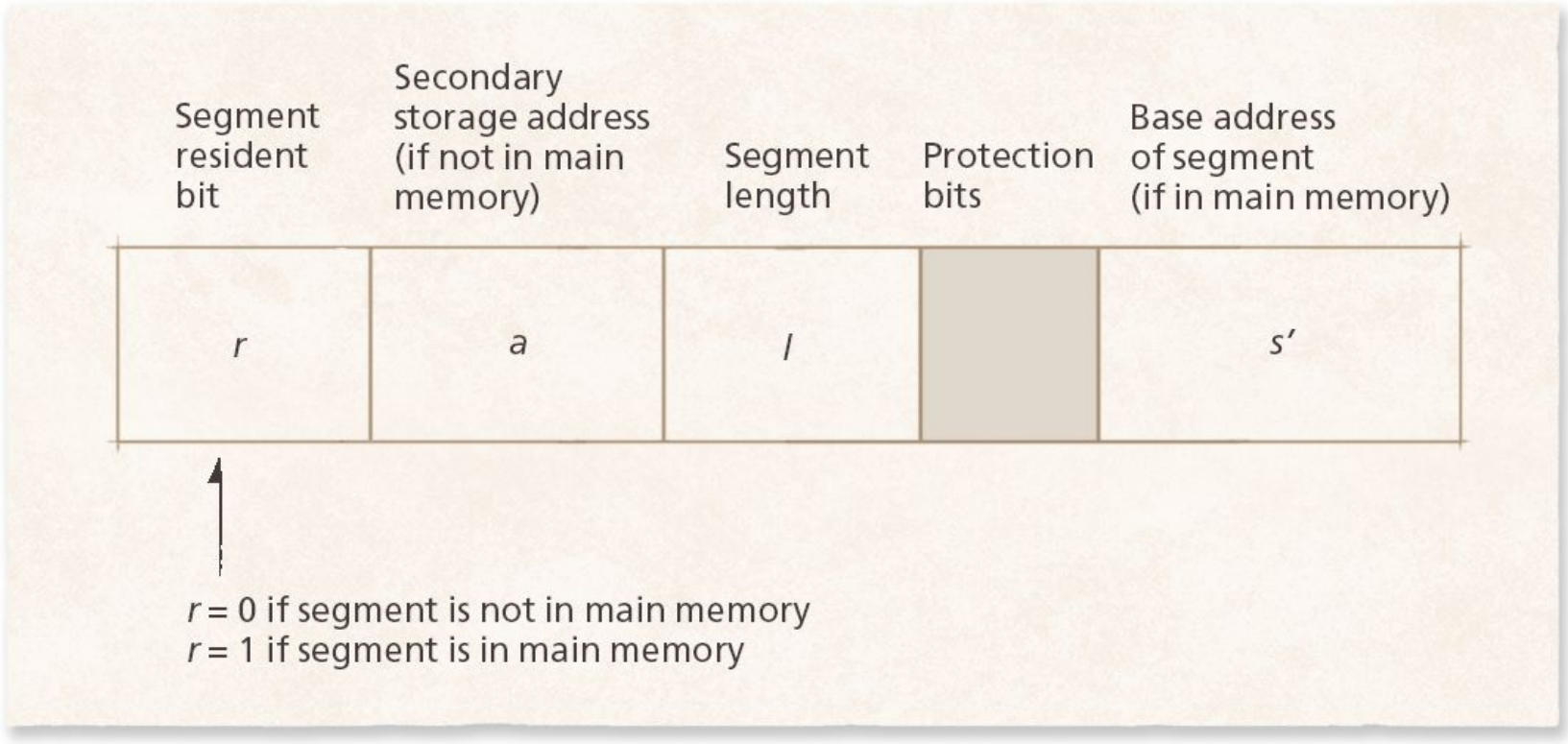
- При таком разбиение ВАП на сегменты виртуальный адрес будет иметь в общем случае следующий вид:

- где *s* – номер сегмента в виртуальной памяти, а *d* – смещение адресуемого байта в пределах сегмента.

Таблица сегментов процесса

- Точно так же, как и при страничной организации используются таблицы страниц, при сегментной организации памяти для преобразования виртуального адреса требуемого элемента в физический адрес используются таблицы сегментов.
- В таблице сегментов, помимо физического адреса начала сегмента, содержится также длина сегмента, биты защиты и прочая служебная информация.

Элемент таблицы сегментов



Перемещение сегментов из вторичной памяти в первичную

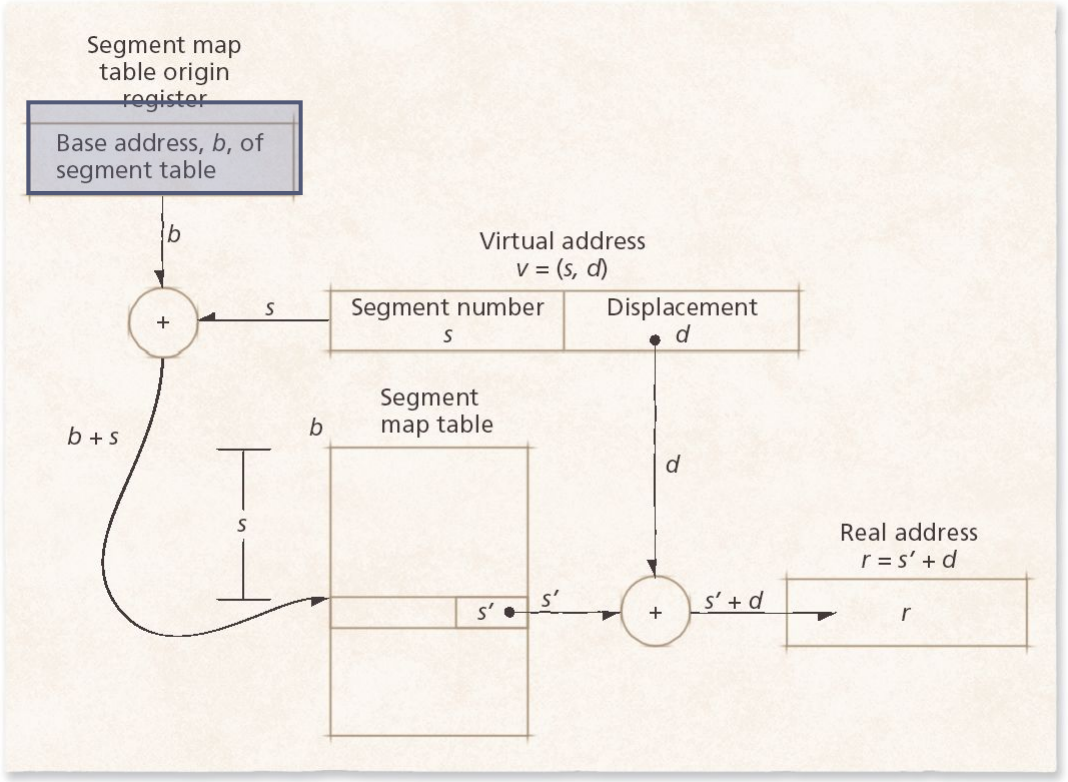
- Система с сегментной организацией функционирует аналогично системе со страничной организацией: время от времени происходят прерывания, связанные с отсутствием нужных сегментов в памяти, при необходимости освобождения памяти некоторые сегменты выгружаются.
- Однако отличительной особенностью сегментной организации является то, что сегменты могут быть разного размера и свободные фрагменты адресного пространства (или сегменты – кандидаты на выгрузку) также могут быть разного размера, поэтому при загрузке каждого из этих сегментов операционная система будет вынуждена выбирать место на загрузку на альтернативной основе.
- Результатом подобной загрузки-выгрузки сегментов может быть фрагментация виртуального адресного пространства.

Сегментное распределение: преобразование ВА в ФА

- 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 ВА в ФА. 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 (номер сегмента *s*, смещение в сегменте *d*). Физический адрес получается путем сложения начального физического адреса сегмента *s'*, найденного в таблице сегментов по номеру и смещения *d*.

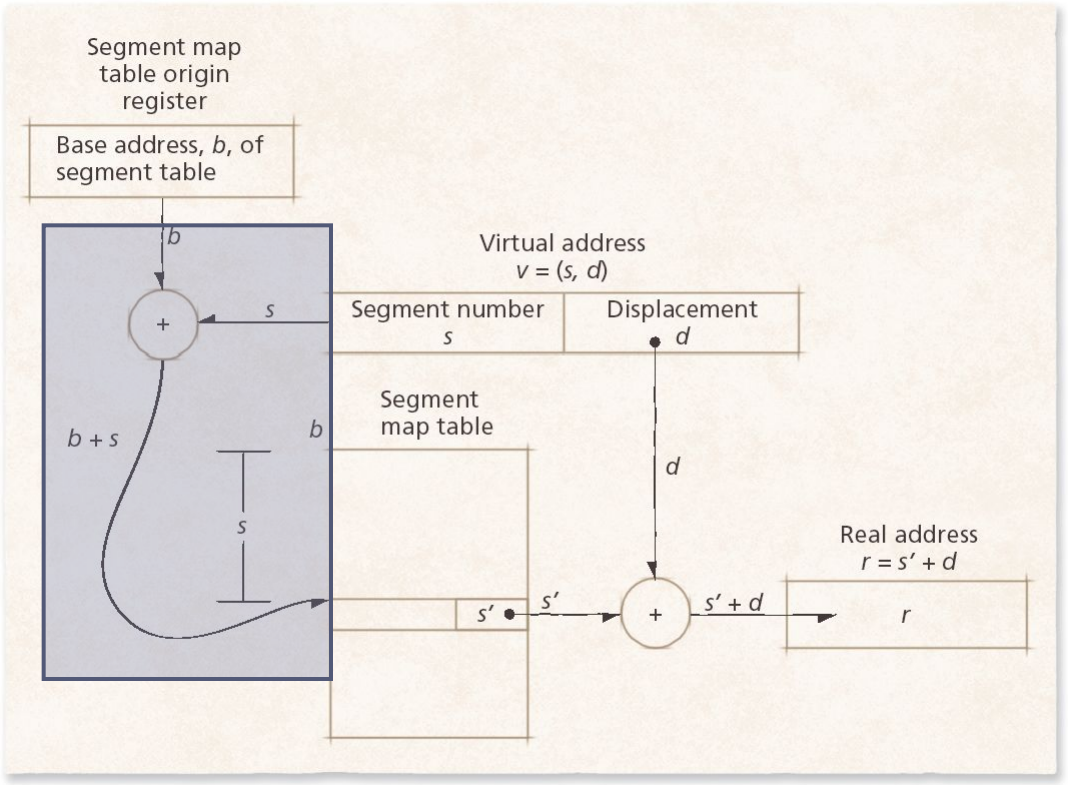
Трансляции адреса в сегментной системе (1)

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



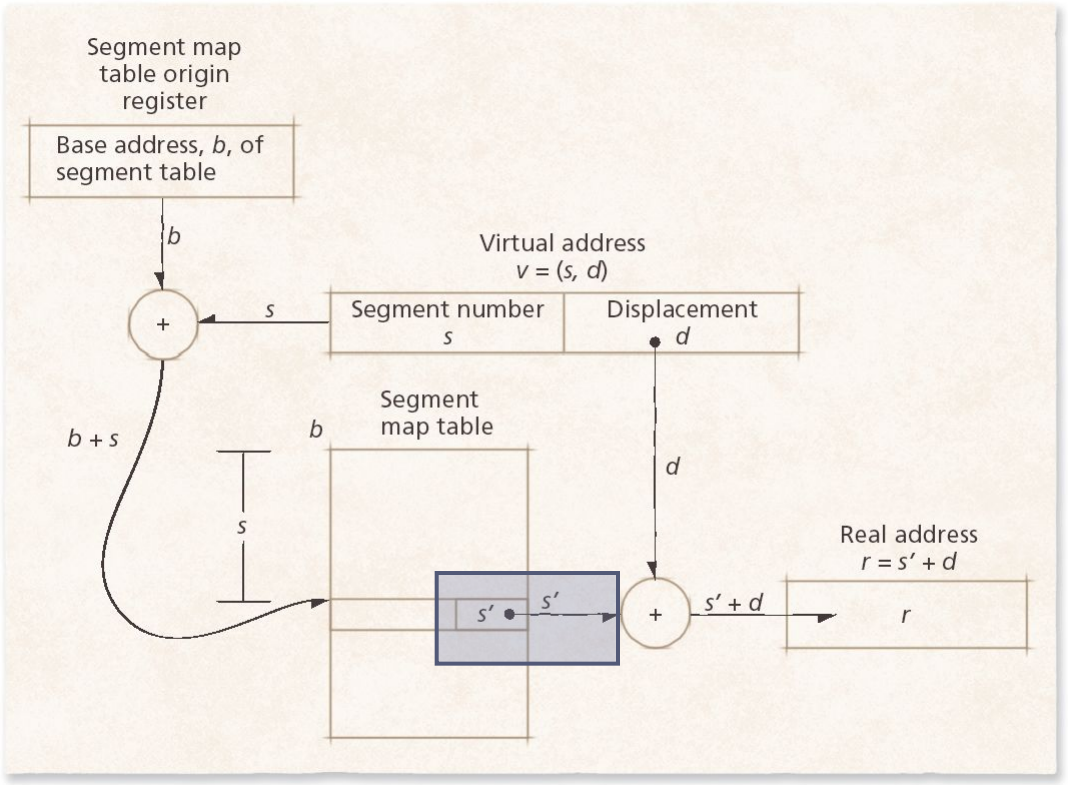
Трансляции адреса в сегментной системе (2)

2. На основании начального адреса таблицы страниц, номера виртуального сегмента страницы и длины записи в таблице сегментов определяется адрес нужной записи.



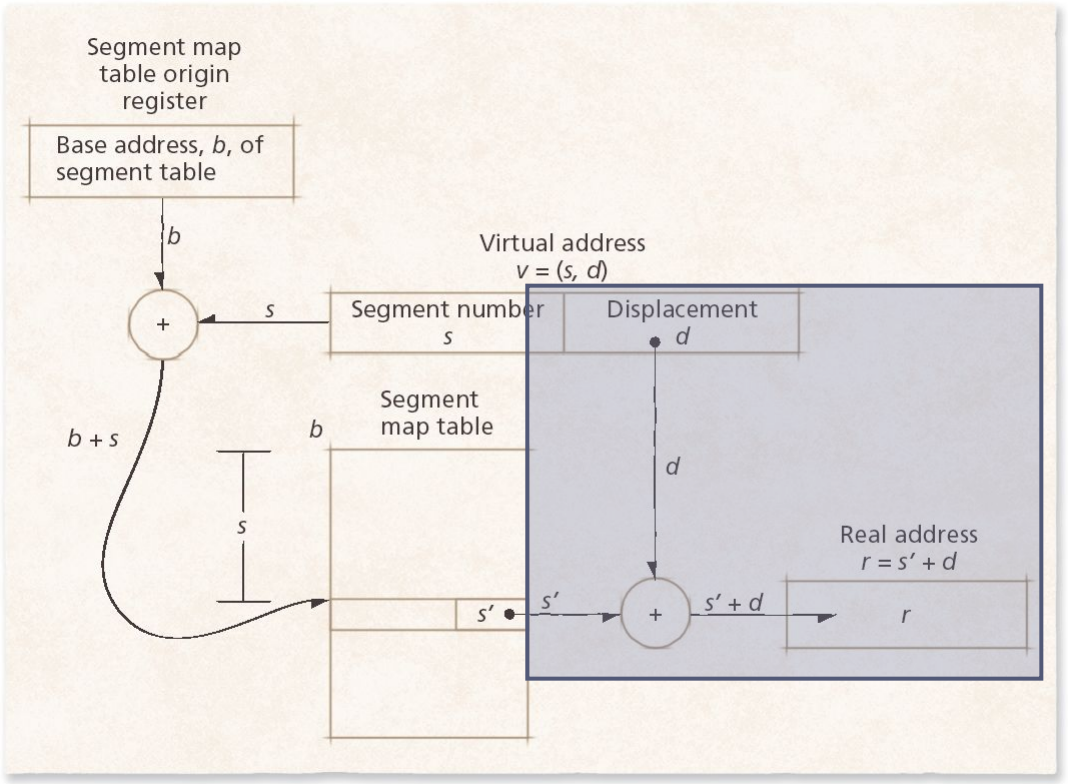
Трансляции адреса в сегментной системе (3)

3. Из этой записи таблицы сегментов извлекается номер физического сегмента.

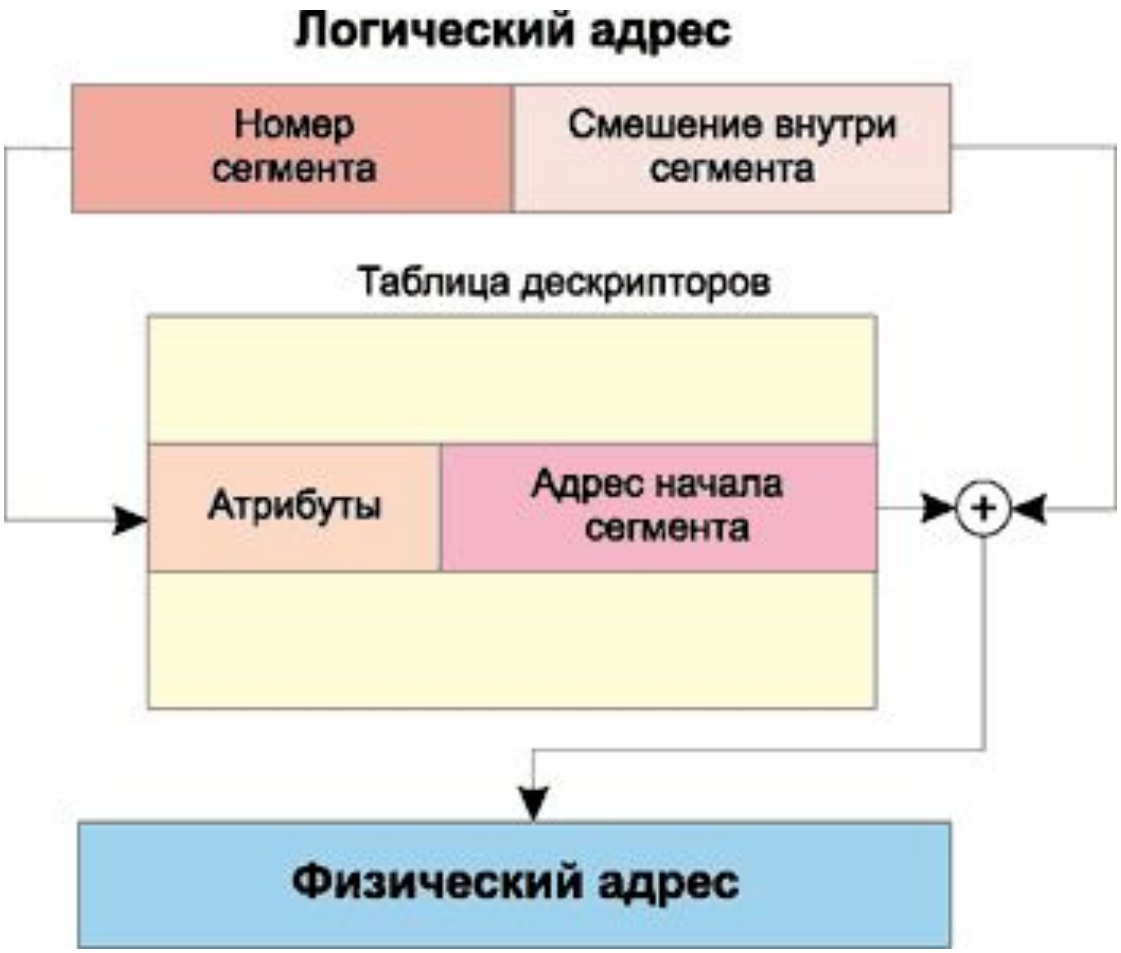


Трансляции адреса в сегментной системе (4)

4. К физическому адресу начала сегмента прибавляется смещение виртуального адреса.

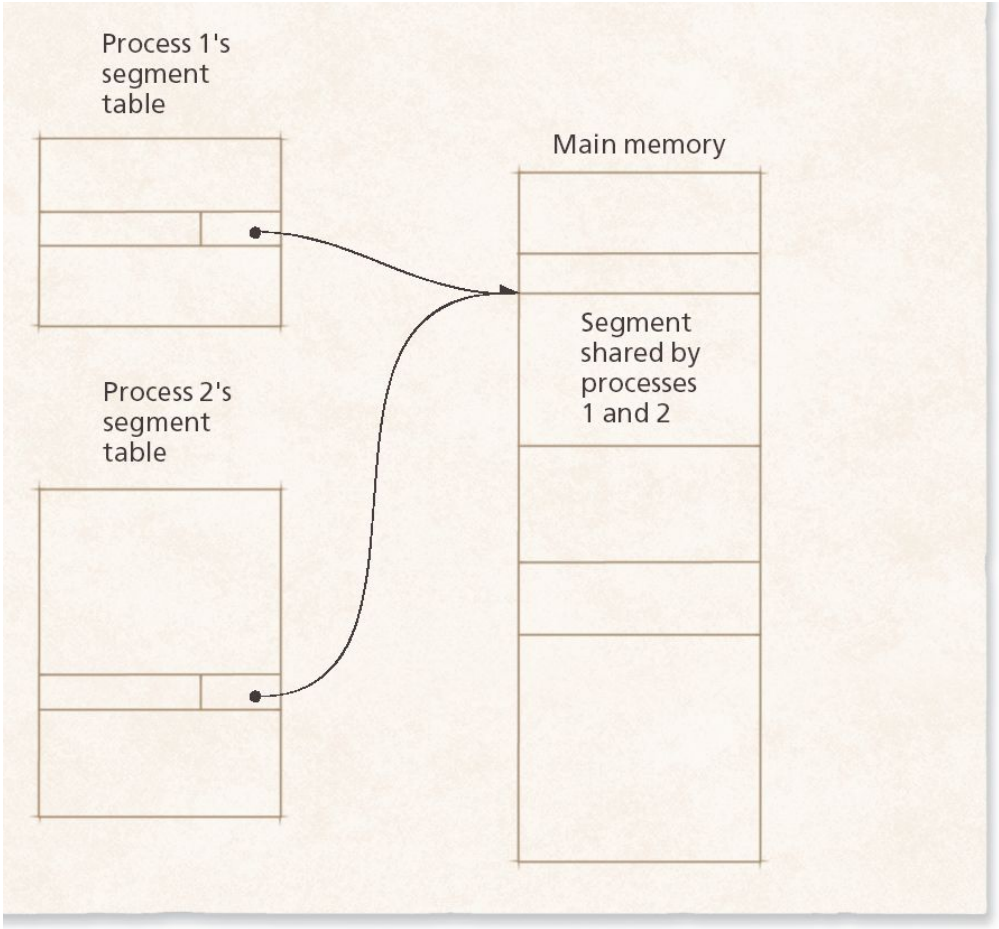


Сегментное распределение: преобразование ВА в ФА



Совместное использование сегментов

- В рамках сегментной модели возможно 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.

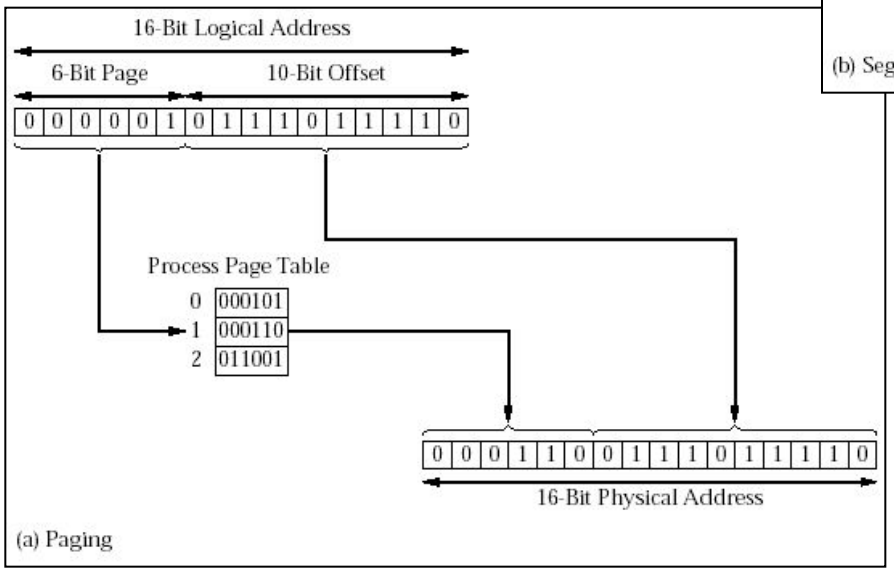


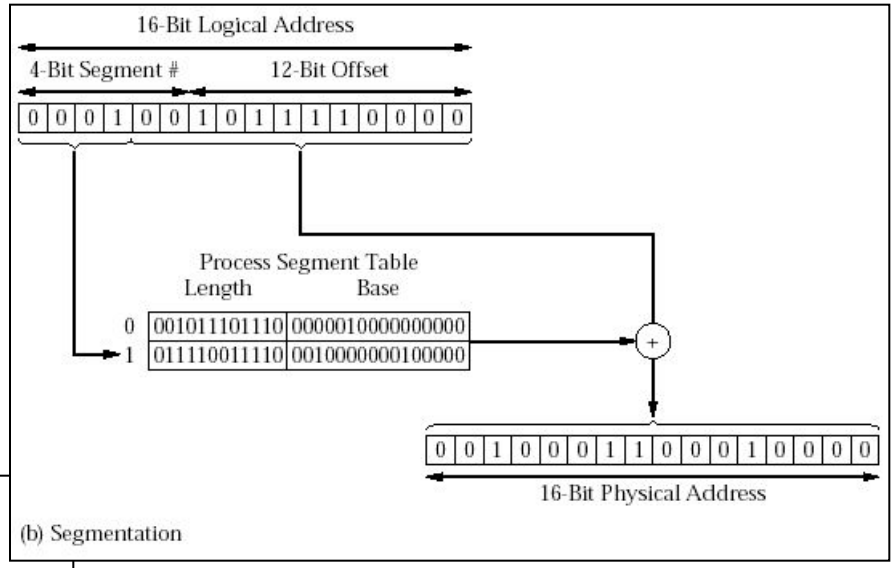
Достоинства и недостатки сегментного распределения

- Более эффективное использование физической (оперативной) памяти.
- Потенциально меньшее количество прерываний по загрузке сегментов (локальность данных в сегменте).
- Более простая возможность защиты данных на уровне сегментов.
- Более медленное (по сравнению со страничным распределением) преобразование ВА в ФА в связи с использованием операции сложения.
- Фрагментация виртуального адресного пространства.
- Сложность реализации свопинга, т.к. сегменты разного размера.

Задание

- Сравните страничное и сегментное преобразования для 16-битной системы.





Методы распределения памяти с использованием дискового пространства

Сегментно-страничное распределение

Сегментно-страничное распределение

- Данный метод представляет собой комбинацию страничного и сегментного распределения памяти и, вследствие этого, сочетает в себе достоинства обоих подходов.
- ВАП 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.

Формат виртуального адреса

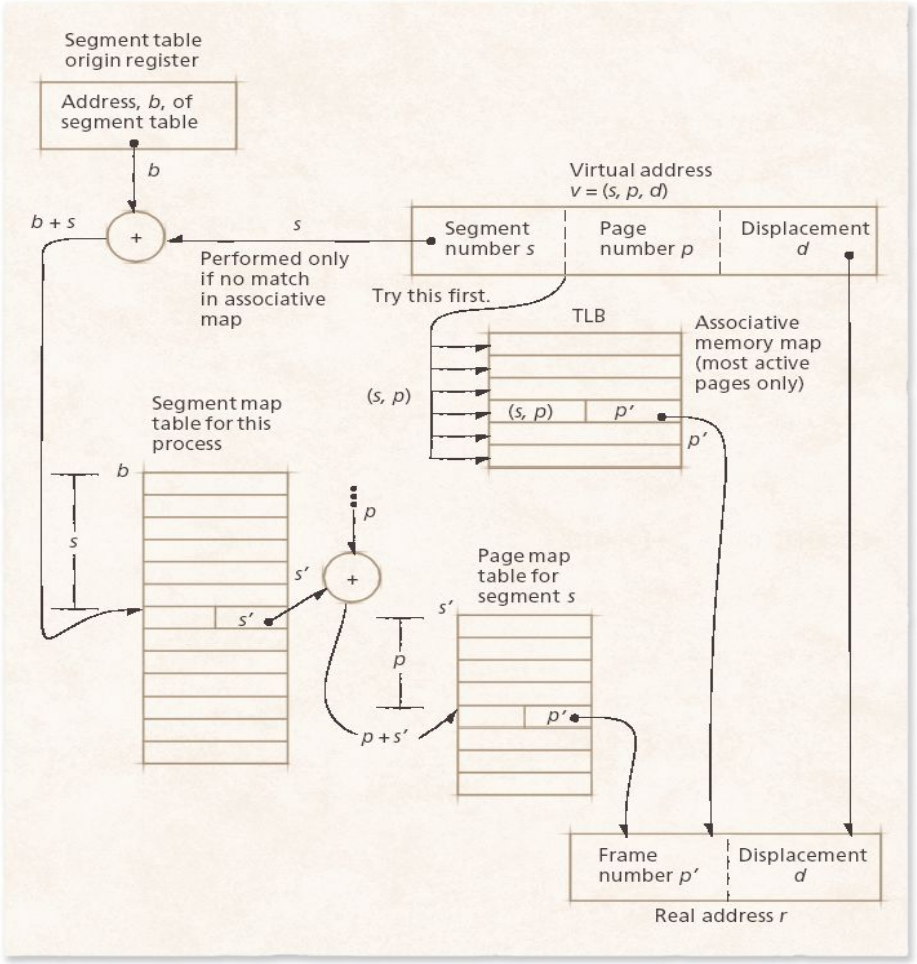
- При таком разбиение ВАП на сегменты виртуальный адрес будет иметь в общем случае следующий вид:

- где *s* – номер сегмента в виртуального памяти, *p* – номер виртуальной страницы процесса в рамках сегмента, а *d* – смещение адресуемого байта в пределах страницы сегмента.

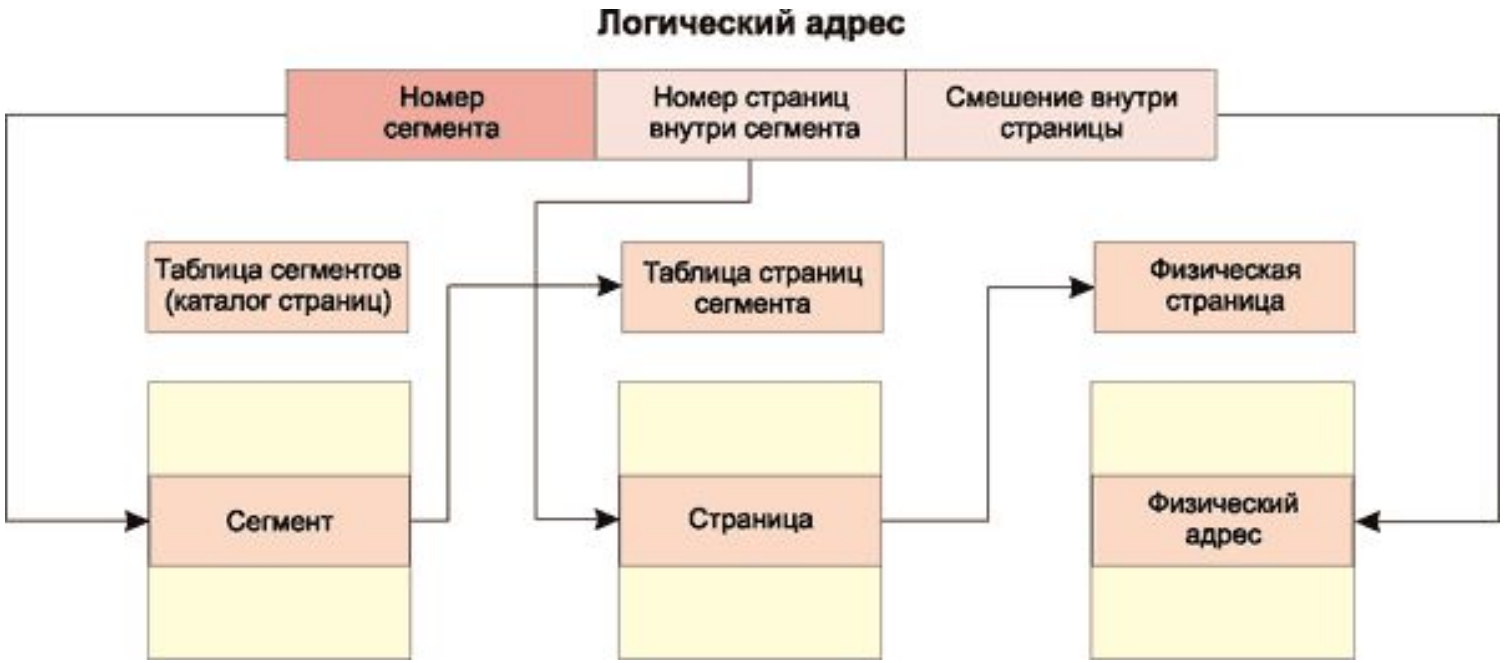
Таблицы сегментов и страниц

- 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.
- Для каждого сегмента создается своя таблица страниц, структура которой полностью совпадает со структурой таблицы страниц, используемой при страничном распределении.

Сегментно-страничное распределение: преобразование ВА в ФА



Сегментно-страничное распределение: преобразование ВА в ФА



Достоинства сегментно-страничного распределения

- Сегментно-страничная виртуальная память сочетает достоинства обоих предыдущих подходов:
  - быстрое преобразование виртуальных адресов в физические;
  - отсутствие больших таблиц страниц;
  - отсутствие фрагментации виртуального адресного пространства;
  - возможность защиты данных на уровне сегментов;
  - простая реализация свопинга.

Вопрос

- Сравните сегментно-страничное преобразование и страничное преобразование с использованием двухуровневых таблиц.