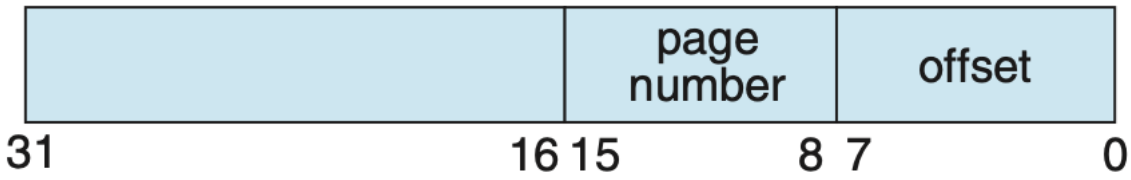
Лабораторная работа №3 “Менеджер виртуальной памяти”

Этот проект состоит из написания программы, которая преобразует логические адреса в физические для виртуального адресного пространства размером 2^{16} = 65 536 байт. Ваша программа должна считать файл, содержащий логические адреса, и, используя TLB и таблицу страниц, преобразовать каждый логический адрес в соответствующий физический адрес и вывести значение байта, хранящегося по преобразованному физическому адресу. Цель лабораторной работы — используя моделирование, понять этапы преобразования логических адресов в физические. Это будет включать в себя устранение ошибок страниц с использованием подкачки по запросу (demand paging), управление TLB и реализацию алгоритма замещения страниц.

Спецификация

Ваша программа считывает файл, содержащий несколько 32-битных целых чисел, представляющих логические адреса. Однако, вам нужно иметь дело только с 16-битными адресами, поэтому вам необходимо замаскировать крайние правые 16 бит каждого логического адреса. Эти 16 бит делятся на (1) 8-битный номер страницы и (2) 8-битное смещение страницы. Таким образом, адреса структурированы следующим образом:



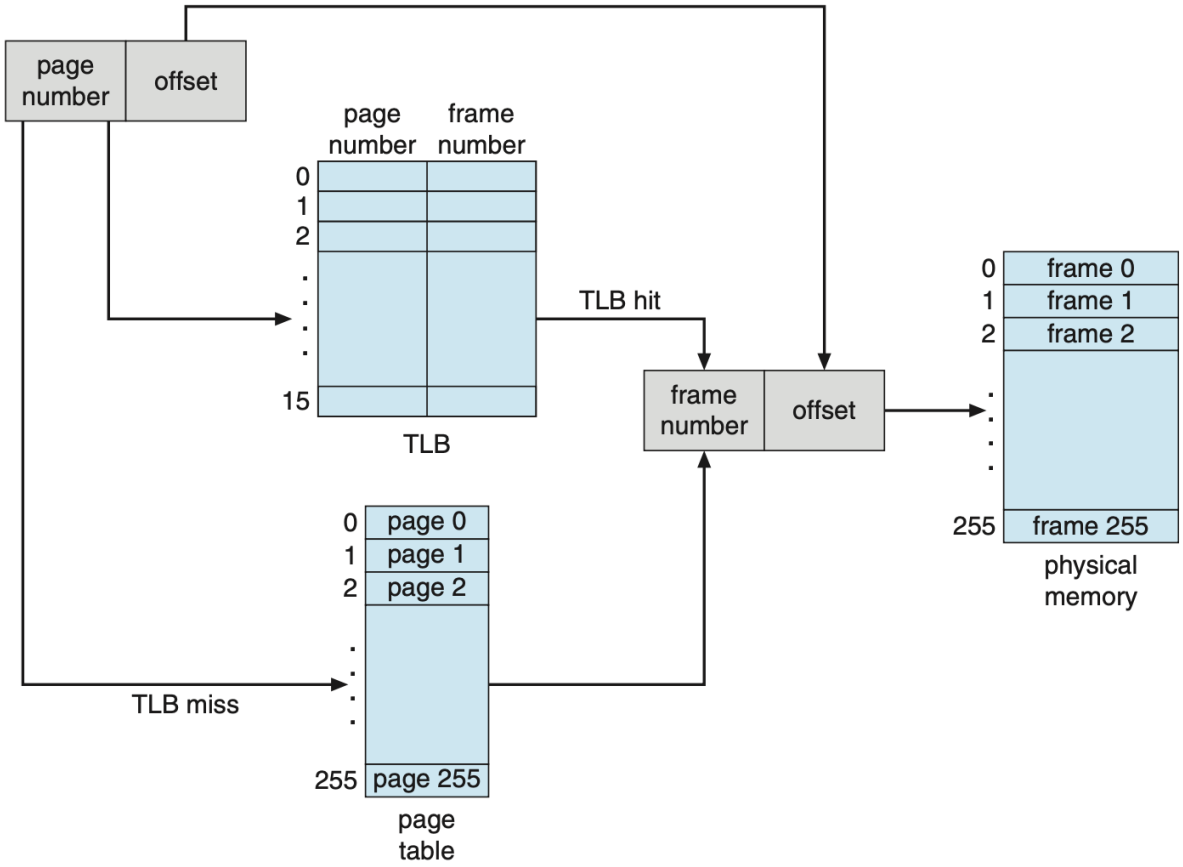
Другие особенности включают следующее:

- 2^8 записей в таблице страниц
- Размер страницы 2^8 байт
- 16 записей в TLB
- Размер фрейма 2^8 байт.
- 256 фреймов
- Физическая память — 65 536 байт (256 фреймов × размер фрейма 256 байт).

Добавим, что ваша программа должна заниматься только чтением логических адресов и преобразованием их в соответствующие физические адреса. Вам не требуется поддерживать запись в логическое адресное пространство.

Трансляция адресов

Программа преобразует логические адреса в физические, используя TLB и таблицу страниц. Сначала номер страницы извлекается из логического адреса и проверяется на наличие в TLB. В случае наличия(hit) в TLB номер фрейма извлекается из TLB. В случае отсутствия(miss) в TLB необходимо обратиться к таблице страниц. В последнем случае либо номер фрейма извлекается из таблицы страниц, либо возникает ошибка страницы. Визуальное представление процесса трансляции адресов:



Обработка ошибок страниц

Программа должна реализовывать подкачку по запросу (demand paging). Резервное хранилище страниц представлено файлом **BACKING STORE.bin**, двоичным файлом размером 65 536 байт. При возникновении ошибки страницы программа должна прочитать 256-байтовую страницу из файла BACKING STORE и сохраните ее в доступном фрейме физической памяти. Например, если логический адрес с номером страницы 15 привел к ошибке страницы, ваша программа прочитает страницу 15 из BACKING STORE (помните, что страницы начинаются с 0 и имеют размер 256 байт) и сохранит ее в фрейме в физической памяти. Как только этот фрейм будет сохранен (и таблица страниц и TLB будут обновлены), последующие обращения к странице 15 будут разрешаться либо TLB, либо таблицей страниц.

Файл **BACKING STORE.bin** нужно рассматривать как файл с произвольным доступом, чтобы вы могли случайным образом искать определенные позиции файла для чтения.

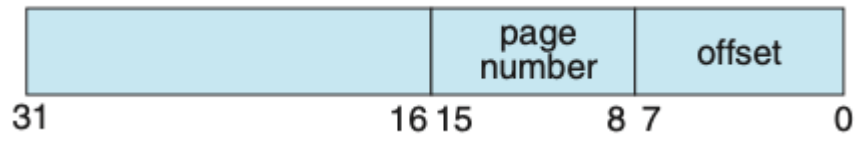
Размер физической памяти такой же, как размер виртуального адресного пространства — 65 536 байт — поэтому вам не нужно беспокоиться о замещение страниц во время ошибки страницы. Если объем физической памяти меньше, то потребуется стратегия замены страниц.

Тестовый файл

Вам предоставлен файл **addresses.txt**, который содержит целочисленные значения, представляющие логические адреса в диапазоне от 0 до 65535 (размер виртуального адресного пространства). Ваша программа откроет этот файл, прочитает каждый логический адрес, преобразует его в соответствующий физический адрес и выведет значение байта с учетом знака(signed byte) по физическому адресу.

С чего начать?

Сначала напишите простую программу, которая извлекает номер страницы и смещение на основе:



для следующих целых чисел:

1, 256, 32768, 32769, 128, 65534, 33153

Самый простой способ сделать это — использовать операторы побитовой маскировки и побитового сдвига. Как только вы сможете правильно установить номер страницы и смещение от целого числа, вы готовы приступить к работе.

Изначально предлагается не использовать TLB, а использовать только таблицу страниц. Вы сможете интегрировать TLB, как только ваша таблица страниц будет работать правильно. Помните, что преобразование адресов может работать без TLB; TLB просто делает это быстрее. Когда вы будете готовы реализовать TLB, помните, что в нем всего шестнадцать записей, поэтому вам нужно будет использовать стратегию замещения при обновлении полного TLB. Для обновления TLB вы можете использовать политику FIFO или LRU.

Как запускать программу?

Программа должна запускаться следующим образом:

./a.out addresses.txt

Программа считывает файл **addresses.txt,** который содержит 1000 логических адресов в диапазоне от 0 до 65535. Программа должна преобразовать каждый логический адрес в физический адрес и определить содержимое подписанного байта, хранящегося по правильному физическому адресу. (Напомним, что в языке C тип данных char занимает один байт памяти, поэтому мы предлагаем использовать значения char.)

Программа должна вывести следующие значения:

1. Транслируемый логический адрес (целочисленное значение считывается из файла address.txt).
2. Соответствующий физический адрес (во что ваша программа преобразует логический адрес).
3. Значение байта со знаком, хранящееся в физической памяти по преобразованному физическому адресу.

Вам также предоставляется файл **correct.txt**, который содержит правильный вывод для значений из файла **addresses.txt**. Используйте этот файл, чтобы определить, правильно ли ваша программа преобразует логические адреса в физические.

Статистика

После завершения ваша программа должна сообщить следующую статистику:

1. Частота ошибок страниц — процент ссылок на адреса, которые привели к ошибкам страниц.
2. Частота попаданий в TLB. Процент адресных ссылок, которые были найдены в TLB.

Поскольку логические адреса в файле **addresses.txt** были сгенерированы случайным образом и не отражают локальность доступа к памяти, не ожидайте высокой частоты попадания(hit rate) в TLB.

Замещение страниц (дополнительное задание)*

До сих пор предполагалось, что физическая память имеет тот же размер, что и виртуальное адресное пространство. На практике физическая память обычно намного меньше виртуального адресного пространства. На этом этапе проекта теперь предполагается использование меньшего физического адресного пространства со 128 страничными фреймами вместо 256. Это изменение потребует изменения вашей программы таким образом, чтобы она отслеживала свободные страничные фреймы, а также реализации политики замены страниц с использованием FIFO или LRU для устранения страничных ошибок при отсутствии свободной памяти.

Материалы к проекту

Лабораторная работа №3