Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное учреждение высшего образования

«Ульяновский государственный технический университет»

Дисциплина «Операционные системы».

**Лабораторная работа №3. Менеджер виртуальной памяти**

**Выполнил**

Студент группы ИВТИИбд-21

Пешков Максим Алексеевич

**Проверил(а):**

Преподаватель Беляев К. С.

Ульяновск 2025

**Тема:** Менеджер виртуальной памяти  
**Цель работы:**

Изучение принципов работы механизма виртуальной памяти, включая:

* трансляцию логических адресов в физические;
* работу с таблицей страниц и TLB (Translation Lookaside Buffer);
* подкачку страниц по требованию (demand paging);
* измерение показателей эффективности: частота TLB-попаданий и ошибок страниц.

**Используемый язык:** C (GCC, Unix/Linux среда)

## **Задание:**

Реализовать транслятор логических адресов в физические с использованием:

* таблицы страниц фиксированного размера (256 записей),
* TLB (Translation Lookaside Buffer) размером 16 записей.

Обработка ошибок страниц с загрузкой данных из бинарного файла BACKING\_STORE.bin.

Считывание логических адресов из файла addresses.txt.

Вывод:

* логического и физического адреса,
* значения из физической памяти,
* статистики: количество TLB-хитов, TLB hit rate, количество ошибок страниц и page fault rate.

### **Теоретические сведения**

#### Виртуальная память

Виртуальная память позволяет программам работать с непрерывным логическим адресным пространством, которое отображается на физическую память с помощью таблицы страниц.

#### Логический адрес

В данной работе логический адрес — 16-битное значение, состоящее из:

* 8-битного номера страницы (page number),
* 8-битного смещения (offset) внутри страницы.

#### Таблица страниц

Это структура в оперативной памяти, которая сопоставляет виртуальные страницы с физическими фреймами. При отсутствии записи в таблице происходит ошибка страницы.

#### TLB (Translation Lookaside Buffer)

TLB — это небольшой, быстрый кэш, расположенный в процессоре. Он хранит недавно использованные отображения виртуальных страниц в физические фреймы для ускорения доступа.

#### Подкачка по требованию

Если нужная страница отсутствует в памяти (ошибка страницы), она загружается из резервного файла BACKING\_STORE.bin.

### **Структура проекта:**

#### main.c

Основной код, реализующий:

* трансляцию адресов;
* TLB (FIFO);
* таблицу страниц;
* подкачку из файла.

#### BACKING\_STORE.bin

Файл с 256 страницами по 256 байт (всего 65 536 байт).

#### addresses.txt

Файл с логическими адресами.

**Реализация:**

#### 1. Извлечение номера страницы и смещения:

Файл: main.c

int logical\_address = ...;

int page\_number = (logical\_address & 0xFF00) >> 8;

int offset = logical\_address & 0x00FF;

#### 2. Поиск в TLB:

for (int i = 0; i < tlb\_count; i++) {

if (tlb[i].page\_number == page\_number) {

frame\_number = tlb[i].frame\_number;

tlb\_hits++;

tlb\_hit = 1;

break;

}

}

#### 3. Обработка page fault:

if (page\_table[page\_number] == -1) {

page\_faults++;

fseek(backing\_store, page\_number \* PAGE\_SIZE, SEEK\_SET);

fread(physical\_memory[next\_free\_frame], sizeof(signed char), PAGE\_SIZE, backing\_store);

page\_table[page\_number] = next\_free\_frame;

frame\_number = next\_free\_frame;

next\_free\_frame++;

}

#### 4. Обновление TLB (по FIFO):

void add\_to\_tlb(int page\_number, int frame\_number) {

tlb[tlb\_index].page\_number = page\_number;

tlb[tlb\_index].frame\_number = frame\_number;

tlb\_index = (tlb\_index + 1) % TLB\_SIZE;

}

### **Вывод**

В ходе лабораторной работы была разработана и реализована упрощённая модель механизма управления виртуальной памятью, включающая:

* **Трансляцию логических адресов в физические** с использованием таблицы страниц;
* **Буфер ассоциативной трансляции (TLB)** для ускоренного поиска отображений страниц;
* **Обработку ошибок страниц (Page Faults)** с подкачкой данных из вторичной памяти (BACKING\_STORE.bin);
* **Анализ производительности** с вычислением количества и доли TLB-хитов и страничных ошибок.

Реализация показала, что **TLB критически важен** в современных системах, так как он снижает количество обращений к таблице страниц, которая сама по себе может находиться в более медленной памяти.

Кроме того, работа с TLB требует внимательной реализации: необходимо правильно учитывать политику замещения (в данной работе применялась FIFO) и обеспечивать синхронность TLB и таблицы страниц.

**Таким образом**, лабораторная работа позволила на практике понять:

* принципы работы виртуальной памяти;
* взаимодействие TLB и таблицы страниц;
* влияние архитектурных решений на производительность системы управления памятью.

Полученные знания важны для понимания низкоуровневых аспектов работы операционных систем и будут полезны при проектировании эффективного программного обеспечения и взаимодействии с аппаратными средствами.