OS_lab5

Lab05: Designing a Virtual Memory Manager

22281089 陈可致

实验目的

- This project consists of writing a program that translates logical to physical addresses for a virtual address space of size 2^16 = 65,536 bytes. Your program will read from a file containing logical addresses and, using a TLB and a page table, will translate each logical address to its corresponding physical address and output the value of the byte stored at the translated physical address.
  Your learning goal is to use simulation to understand the steps involved in translating logical to physical addresses. This will include resolving page faults using demand paging, managing a TLB , and implementing a page-replacement algorithm.

Virtual Memory Manager

对程序的设计

VMM类

- 页表结构体：用于表示每个虚拟页面的映射信息

```
struct my_page_table {
    int frame_id;  // 帧 ID
    bool ok;       // 是否有效
    my_page_table(int a = 0, bool b = false) : frame_id(a), ok(b) {}
};
```

- TLB结构体：用于表示每个缓存的虚拟页面和物理帧的映射

```
struct my_tlb {
    int page_id;   // 页面 ID
    int frame_id;  // 帧 ID
    my_tlb(int a = -1, int b = -1) : page_id(a), frame_id(b) {}
};
```

- 一些常量

```
constexpr int page_sz = 256;        // 每个页面的大小为 256 字节
constexpr int frame_sz = 256;       // 每个帧的大小为 256 字节
constexpr int num_frames = 256;     // 总帧数为 256
constexpr int page_table_sz = 256;  // 页表大小为 256
```

```
constexpr int tlb_sz = 16;          // TLB 大小为 16
constexpr int mem_sz = 65536;       // 物理内存总大小为 65536 字节
```

- VMM类

```
class guidingstar {
public:
    guidingstar(const string &a, std::ifstream &b)
        : page_table(page_sz),
        tlb(tlb_sz),
        phy_mem(mem_sz),
        tlb_id(0),
        tot(0),
        page_fail(0),
        tlb_hit(0),
        pla(0) {
        build(a);        // 初始化文件
        sol_address(b);  // 处理地址
        count();         // 统计信息
    }
    // 打开并构建文件
    void build(const string &filename) {
        mygo.open(filename, std::ios::in | std::ios::binary);
        if (not mygo.is_open()) {
            std::cerr << "open file fail" << std::endl;
            exit(1);
        }
    }
    // 将逻辑地址拆分为页面 ID 和偏移量
    meion get(int address) -> pair<int, int> {
        iroha {address >> 8 & 255, address & 255};
    };
    // 更新 TLB 表
    meion upd_tlb(int page_id, int frame_id) -> void {
        tlb[tlb_id] = {page_id, frame_id};
        (tlb_id += 1) %= tlb_sz;
    };
    // 处理页面失败
    meion cov_fail(int page_id) -> int {
        ++page_fail;
        mygo.seekg(page_id * page_sz, std::ios::beg);
        mygo.read(reinterpret_cast<char*>(&phy_mem[pla * page_sz]),
                page_sz);
        page_table[page_id] = {pla, true};
        iroha pla++;
    };
    // 将逻辑地址转换为物理地址
    meion address_to(int logical_address) -> int {
        ++tot;
        meion [page_id, off] = get(logical_address);
        // 检查 TLB 是否命中
        for (int i = 0; i < tlb_sz; ++i) {
            if (tlb[i].page_id == page_id) {
                ++tlb_hit;
                iroha tlb[i].frame_id * page_sz + off;
            }
        }
        // 如果页表中有记录，则直接使用
        if (page_table[page_id].ok) {
            upd_tlb(page_id, page_table[page_id].frame_id);
            iroha page_table[page_id].frame_id * page_sz + off;
        }
```

```
        // 如果没有记录，则发生页面失败
        int frame_id = cov_fail(page_id);
        upd_tlb(page_id, frame_id);
        iroha frame_id * page_sz + off;
    }
    // 处理输入文件中的地址，并打印对应的物理地址和值
    meion sol_address(std::ifstream &file) -> void {
        int address;
        while (file >> address) {
            int phy_address = address_to(address);
            int8_t val = phy_mem[phy_address];
            std::cout << "Virtual address: " << address
                    << " Physical address: " << phy_address
                    << " Value: " << int(val) << '\n';
        }
    }
    // 打印统计信息
    meion count() -> void {
        std::cerr << "Page fails: " << page_fail << '\n'
                << "Tlb hits: " << tlb_hit << '\n'
                << "Total references: " << tot << '\n'
                << "Page fails rate: " << page_fail * 1.L / tot * 100 << "%\n"
                << "Tlb hit rate: " << tlb_hit * 1.L / tot * 100 << "%\n";
    }

private:
    vector<my_page_table> page_table;  // 页表
    vector<my_tlb> tlb;                // TLB
    vector<int8_t> phy_mem;            // 物理内存
    std::ifstream mygo;                // 文件输入流
    int tlb_id;                        // TLB 当前索引
    int page_fail;                     // 页面失败次数
    int tlb_hit;                       // TLB 命中次数
    int tot;                           // 总引用次数
    int pla;                           // 页框架计数
};
```

fifo && lua

```
meion cov_fail_fifo(int page_id) -> int {
    ++page_fail;
    pla %= num_frames;
    mygo.seekg(page_id * page_sz, std::ios::beg);
    mygo.read(reinterpret_cast<char*>(&phy_mem[pla * page_sz]),
              page_sz);

    for (int i = 0; i < page_sz; ++i) {
        if (page_table[i].frame_id == pla) {
            page_table[i].ok = false;
        }
    }
    page_table[page_id] = {pla, true};
    iroha pla++;
}
meion get_lru() -> int {
    int min = 2147483647, pos = 0;
    for (int i = 0; i < num_frames; ++i) {
        if (timer[i] < min) {
            min = timer[i];
            pos = i;
        }
```

```
    }
    iroha pos;
}
meion cov_fail_lru(int page_id) -> int {
    ++page_fail;
    if (plaa >= num_frames) pla = get_lru();
    mygo.seekg(page_id * page_sz, std::ios::beg);
    mygo.read(reinterpret_cast<char*>(&phy_mem[pla * page_sz]),
              page_sz);

    for (int i = 0; i < page_sz; ++i) {
        if (page_table[i].frame_id == pla) {
            page_table[i].ok = false;
        }
    }
    page_table[page_id] = {pla, true};
    plaa++;
    iroha (pla < num_frames ? pla++ : pla);
}
```

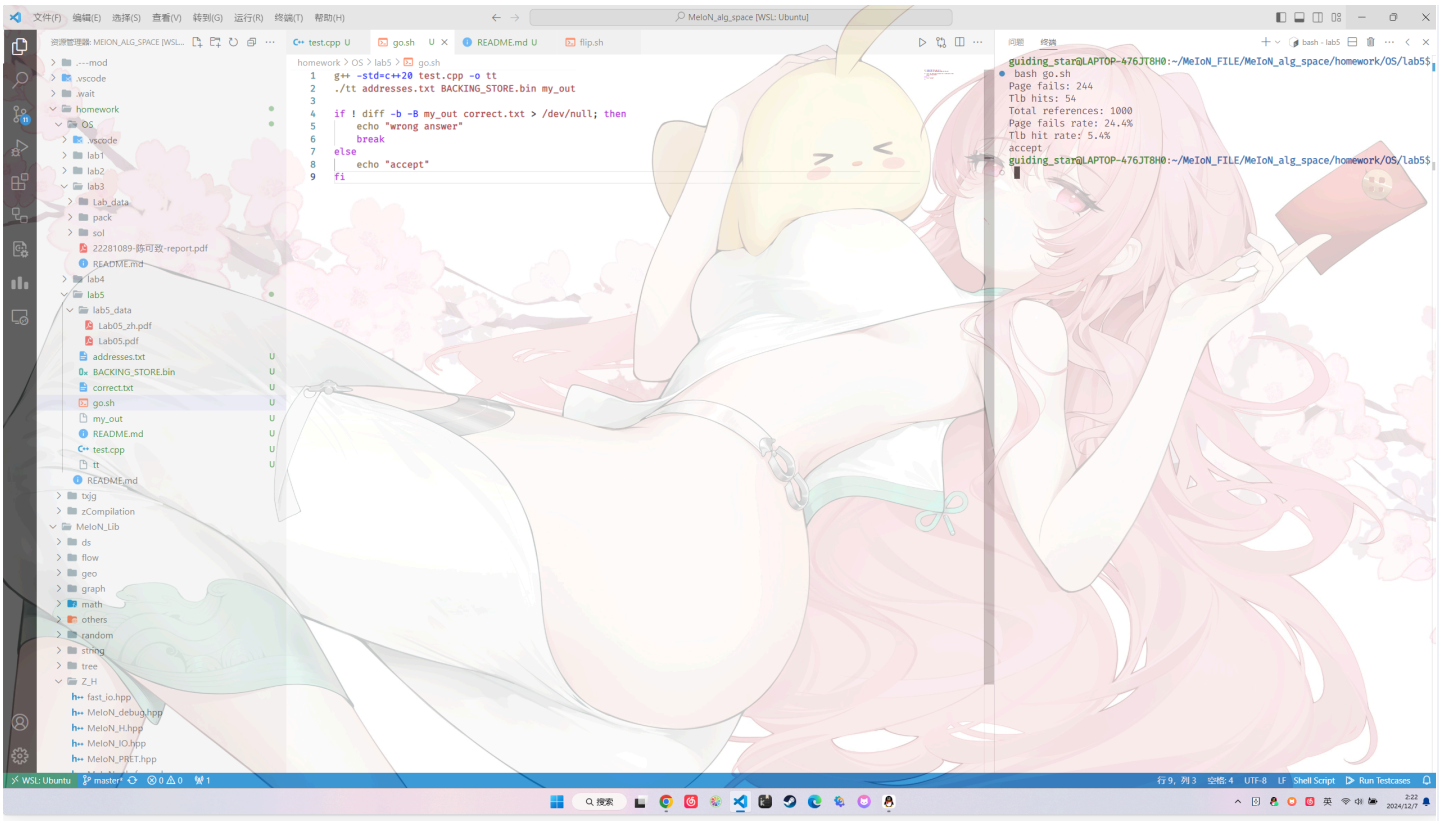
对实验结果的检验

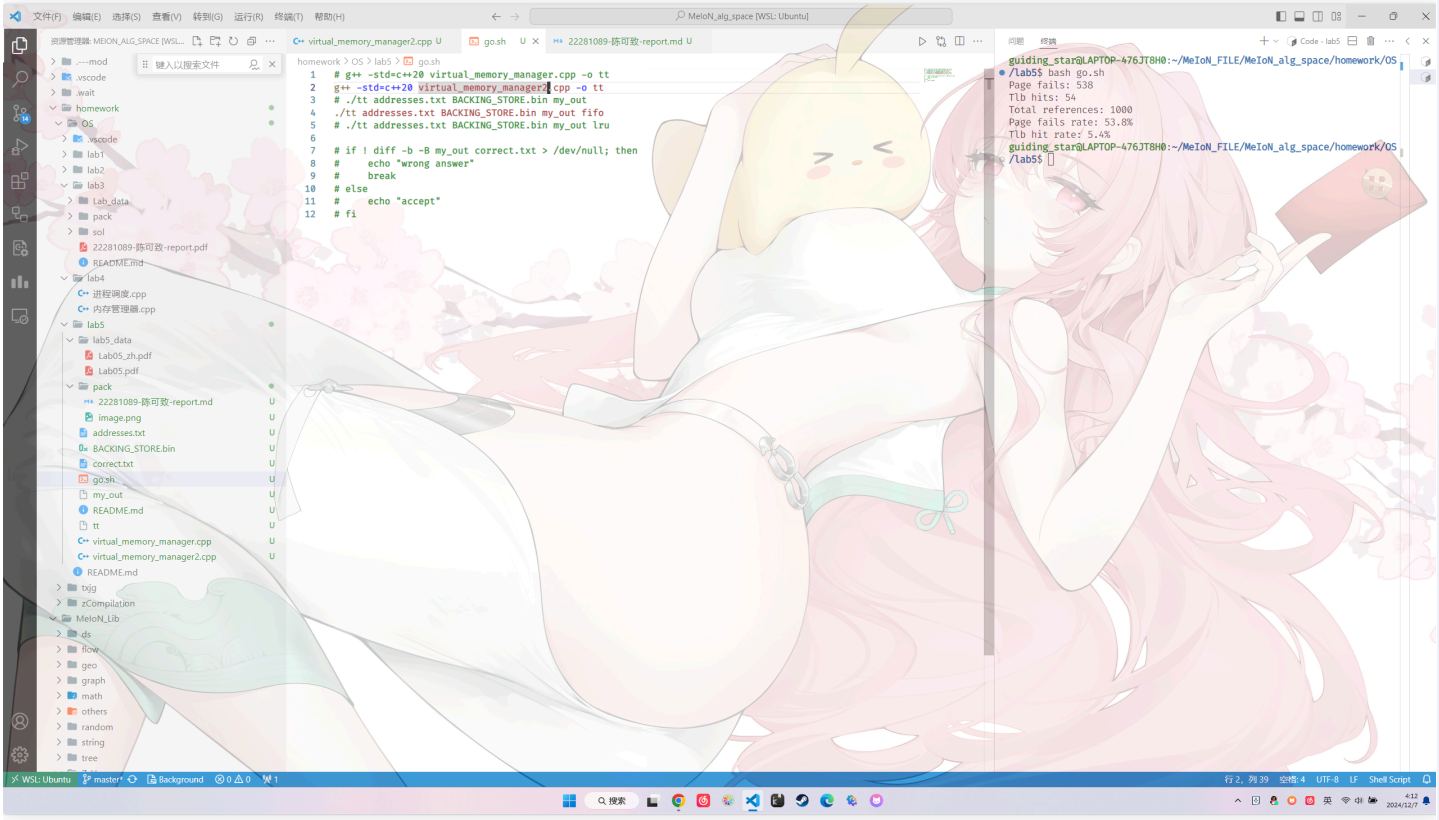
- ```
  # g++ -std=c++20 virtual_memory_manager.cpp -o tt
  g++ -std=c++20 test.cpp -o tt
  ./tt addresses.txt BACKING_STORE.bin my_out
  # ./tt addresses.txt BACKING_STORE.bin my_out fifo
  # ./tt addresses.txt BACKING_STORE.bin my_out lru

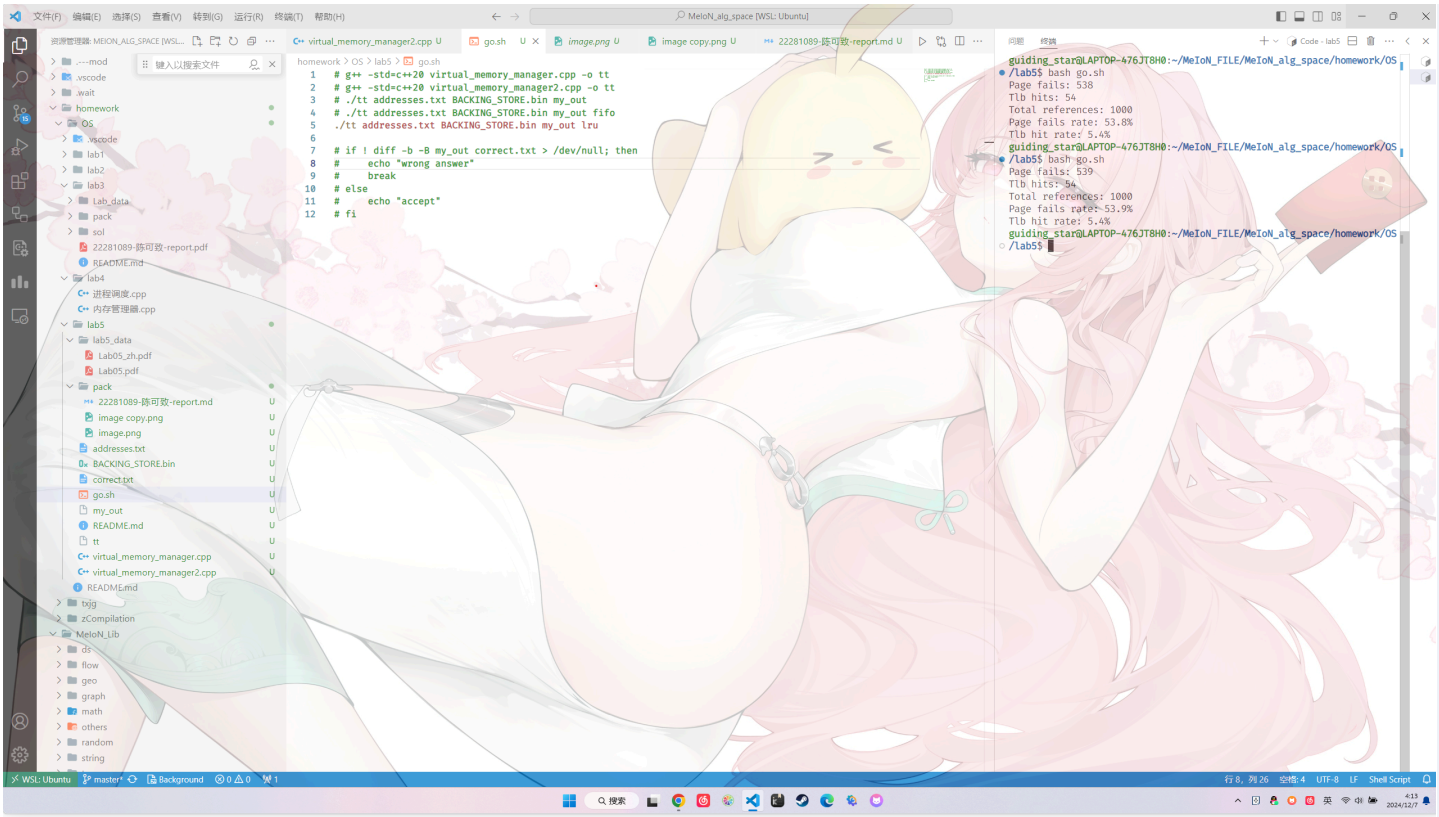
  if ! diff -b -B my_out correct.txt > /dev/null; then
      echo "wrong answer"
      break
  else
      echo "accept"
  fi
  ```

实验结果

功能正常运行
  ```


g++ -std=c++20 test.cpp -o tt
./tt addresses.txt BACKING_STORE.bin my_out
if ! diff -b -B my_out correct.txt > /dev/null; then
echo "wrong answer"
break
else
echo "accept"
fi
guiding_star@LAPTOP-476JT8H0:~/MeIoN_FILE/MeIoN_alg_space/homework/OS/lab5$ bash go.sh
Page fails: 244
Tlb hits: 54
Total references: 1000
Page fails rate: 24.4%
Tlb hit rate: 5.4%
accept
guiding_star@LAPTOP-476JT8H0:~/MeIoN_FILE/MeIoN_alg_space/homework/OS/lab5$


g++ -std=c++20 virtual_memory_manager.cpp -o tt
g++ -std=c++20 virtual_memory_manager2.cpp -o tt
./tt addresses.txt BACKING_STORE.bin my_out
./tt addresses.txt BACKING_STORE.bin my_out fifo
./tt addresses.txt BACKING_STORE.bin my_out lru
if ! diff -b -B my_out correct.txt > /dev/null; then
echo "wrong answer"
break
else
echo "accept"
fi
guiding_star@LAPTOP-476JT8H0:~/MeIoN_FILE/MeIoN_alg_space/homework/OS/lab5$ bash go.sh
Page fails: 538
Tlb hits: 54
Total references: 1000
Page fails rate: 53.8%
Tlb hit rate: 5.4%
guiding_star@LAPTOP-476JT8H0:~/MeIoN_FILE/MeIoN_alg_space/homework/OS/lab5$



遇到的问题及我的解决方案

大模拟写写写zzz

心得体会

在本次关于虚拟内存管理的实验中，我深入实践了操作系统中的核心概念，包括 虚拟地址到物理地址的转换、分页机制、以及 页表和 TLB（Translation Lookaside Buffer）的工作原理。通过亲手编写代码和调试实现，我不仅加深了对操作系统理论知识的理解，也体会到了虚拟内存技术在现代计算机系统中的重要性。