实验三 分页式内存管理

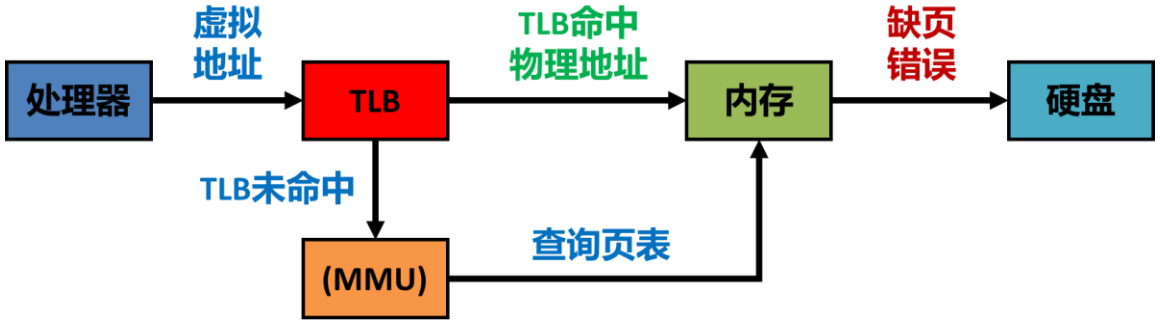
1、实验目的

- 了解分页式内存管理机制
- 了解 TLB 机制，虚拟地址如何转换为物理地址，多级页表管理，页面错误的处理方法

2、实验内容

1.1 任务描述

本次实验需要仿真下图中所示的系统。实验需要构建一个用户级页表，通过二级页表实现虚拟地址到物理地址的转换；设计并实现一个 TLB 缓存以降低转换开销；出现缺页错误时，模拟操作系统中页面错误的处理机制进行页面替换。为简化问题，本次实验使用支持 4GB 地址空间的 32 位地址空间。测试时将调整物理内存大小、页面大小和 TLB 条目数量。



1.2 虚拟内存系统

本实验的目标是实现 myMalloc()和 myFree()完成内存管理，深入思考虚拟页如何转换为物理页以及如何管理它们。具体需要实现的接口函数如下：

`myMalloc()`

功能说明：此函数输入为要分配的字节数，返回虚拟地址。由于使用 32 位虚拟地址空间，需自行管理地址分配。必须使用**位图**（bitmap）跟踪已分配和空闲的页面，位图需高效实现（每页仅占用 1 位），避免内存浪费。为简化逻辑，每次分配根据请求大小分配一个或多个页。例如：

- 连续两次调用 `myMalloc(1 byte)` 将为每次调用各分配一个独立的页（虽然每次只申请 1byte）
- `myMalloc(4097 bytes)` 在 4KB 页面大小下将分配 2 个页

`myFree()`

功能说明：此函数输入为虚拟地址和字节数，释放从该地址开始的页。例如，在 4KB 页面大小下，`myFree(0x1000, 5000)` 将释放从 0x1000 开始的两个页。需确保不释放未分配的页。仅当所有页成功释放时返回成功。当释放未分配的页时（如同一个地址连续调用 myFree 两次），在屏幕上输出“Segmentation Fault”，但不需终结程序或做任何错误处理。

myWrite()

功能说明：此函数输入为虚拟地址、值指针和值大小，直接将数据复制到物理地址。调用时需验证虚拟地址的有效性，如虚拟地址无效，在屏幕上输出“ERROR： Writing to unallocated address”，但不需终结程序或做任何错误处理。

myRead()

功能说明：此函数输入为虚拟地址、值指针和值大小，从虚拟地址读取数据到缓冲区。需先检查 TLB 中是否存在转换，如果没有转换则需要处理 TLB 缺失。调用时需验证虚拟地址的有效性，如虚拟地址无效，在屏幕上输出“ERROR： Reading from unallocated address”，但不需终结程序或做任何错误处理。

除上述接口函数外，本实验还需要实现一些功能函数。下面是可能会用到的重要函数，其他函数可根据具体实现细节自行添加。

initMemoryAndDisk()

功能说明：此函数负责通过 Linux 中的 mmap 或 malloc 分配内存缓冲区，以模拟物理内存和硬盘。物理内存指的是通过 mmap()或 malloc()分配的大块连续内存区域，为页表和内存管理器提供物理内存的假象。硬盘也可通过相同方式分配，与物理内存分开管理。允许使用 malloc 分配其他数据结构以管理虚拟内存。

translate()

功能说明：此函输入为一级页表地址、虚拟地址，返回对应的物理地址。需实现两级页表，为简化实现，可假设两级页表所用位数相同。例如，在 4KB 页大小配置中，每级使用 10 位，剩余 12 位作为偏移量。对于其他页大小 X，偏移量占用 log2(X)位，剩余位分为两级（评分时只会测试 log2(X)为偶数的情况）。

pageMap()

功能说明：此函数遍历页目录，检查虚拟地址是否已存在映射。若不存在，则添加新条目。需在 myMalloc()中调用此函数以添加页表项。

pageFault()

功能说明：此函数处理页面缺失错误，进行物理页面替换与新页面加载。本实验不指定替换机制，但需要在报告中说明你实现了哪种替换机制。

示例代码与工具：可参考 sample.c 学习位操作，包括：

1）从 32 位数提取高位 N 位。

2）提取中间 M 位。

3）设置字符数组中的第 K 位。

4）读取字符数组中的第 K 位。

注意：请勿修改以下函数的名称以及参数：myMalloc()，myFree()，myWrite()，myRead()。代码必须正确实现临界区保护，评分时将使用多线程程序进行测试。

1.3 TLB 实现

本实验需要实现全相联 TLB，缓存虚拟页号到物理页号的转换。具体细节如下：

1）基本逻辑

- 初始化页表时创建全相联 TLB
- 新分配的页在 TLB 中无转换条目，需通过 `addTLB()`添加
- 在 `translate()`中，先通过 `checkTLB()`检查 TLB 中是否存在转换。若 TLB 命中，则无需遍历页表。若 TLB 未命中但页表存在转换，则通过 `addTLB()`添加新条目。
- 若 TLB 未命中且页表中不存在转换（即页面缺失），调用 `pageFault()`处理缺页错误并相应地更新页表和 TLB。

2）TLB 条目数

TLB 条目数由 `defines.h` 中的 `TLB_SIZE` 定义，代码需支持任意条目数。测试时，助教将修改 `defines.h`，对不同的 TLB 大小进行测试。

3）TLB 条目内容

每个 TLB 条目需至少包含有效位、虚拟页号、物理页号。其他内容可根据具体实现酌情添加。

4）TLB 替换策略

当 TLB 满时，需替换旧条目。替换逻辑需在 `addTLB()`中实现，本实验不指定替换机制，但需要在报告中说明你实现了哪种替换机制。

5）输出要求

需通过 `printTLBStats()`报告 TLB 访问次数和缺失率。缺失率=(缺失次数/访问次数)*100%。

注意：代码必须正确实现临界区保护，评分时将使用多线程程序进行测试。

3、建议步骤

1）设计内存管理库的基础数据结构
2）实现 `initMemoryAndDisk()`、`translate()`、`pageMap()`、`pageFault()`
3）实现 `myMalloc()`和 `myFree()`
4）实现 `myWrite()`和 `myRead()`，使用基准程序测试代码
5）加入 TLB 功能

4、编译与基准测试

请使用提供的 Makefile 编译，32 位代码需使用-m32 标志编译。矩阵乘法基准测试用于验证虚拟内存函数。需先编译虚拟内存库代码生成库文件，接下来进入 benchmark 文件夹编译基准测试文件。如果虚拟内存库实现不正确，则可能导致基准测试运行结果错误。

5、实验报告撰写要求

实验报告内容与形式：

1）实验报告中请阐述各虚拟内存函数的详细实现逻辑，你在实验过程中遇到的问题和思考。

2）提供基准测试输出和 TLB 缺失率。

3）阐述如何支持不同页面大小，分析不同页面大小下 TLB 访问次数与缺失率。

4）在实验报告最后以附录的形式分别粘贴 my_vm.h 和 my_vm.c 代码。**助教会对所有的实验报告进行查重，请各组独立完成编码与实验报告。我们对抄袭零容忍。**

6、实验报告与代码提交要求（会影响最后评分，请务必按格式要求提交）

提交内容，只提交三个文件：

1）**my_vm.h** 文件

2）**my_vm.c** 文件

3）**实验报告 word 文档**，命名方式：**OSLab3-成员 1 学号姓名-成员 2 学号姓名.docx**

例如：OSLab3-B22035678 张三- B22035679 李四.docx

将以上三个文件打成一个压缩包，命名方式：**OSLab3-成员 1 学号姓名-成员 2 学号姓名.zip**。由班长或学委收集，统一发给任课老师。

7、实验评分标准

给分点	分数
提交的代码能够正确编译，所有给出的测试程序运行结果正确	10%
用于评分的测试程序运行结果正确（未提供给大家）	30%
虚拟内存管理相关函数(myMalloc, myFree, myWrite, myRead 以及其他功能函数)实现正确	20%
TLB 相关函数功能正确	10%
代码注释详细、正确，实验报告文档内容详实	30%
总计	**100%**