lab2-2 Exam

创建并切换分支

```
git checkout lab2
git add .
git commit -m "xxxxx"
git checkout -b lab2-2-exam
```

题目描述

请你对现有的虚拟内存管理机制进行增量开发，在**完整**保留现有功能的同时，实现物理地址的反向查询函数：

物理地址的反向查询函数 `inverted_page_lookup`

- 函数原型为：`int inverted_page_lookup(Pde *pgdir, struct Page *pp, int vpn_buffer[])`
- 调用此函数后，找到页目录 `pgdir` 对应的整个两级页表中，映射到 `pp` 对应物理页面的所有虚拟页号，将其按**升序**放入 `vpn_buffer` 中，并返回虚拟页号的个数。特别地，如果不存在这样的虚拟页号，直接返回 0。

注意事项

你需要先在 `include/pmap.h` 中加入如下函数定义：

```
int inverted_page_lookup(Pde *pgdir, struct Page *pp, int vpn_buffer[]);
```

之后再在 `mm/pmap.c` 中实现这个函数。

若你需要修改 `Page` 结构体的定义，请你保证 `sizeof(struct Page)` 不超过 256。

评测逻辑

评测过程中，**我们会将所有的 `Makefile` 文件、`include.mk` 以及 `init/init.c` 替换为 lab2 初始配置**，接着将 `init/init.c` 中的 `mips_init` 函数改为如下形式：

```
void mips_init(){
        mips_detect_memory();
        mips_vm_init();
        page_init();

        inverted_page_lookup_test();

        *((volatile char*)(0xB0000010)) = 0;
}
```

最后的 `*((volatile char*)(0xB0000010)) = 0;` 会终止 Gxemul 模拟器的运行，避免占用评测资源。

`inverted_page_lookup_test()` 是在评测过程中新添加到 `init/init.c` 的函数，其形式为：

```
static void inverted_page_lookup_test(){
        struct Page *pp;
        page_alloc(&pp);
        Pde *pgdir = (Pde*)page2kva(pp);

        // test
}
```

`// test` 实际包含以下四种操作：

- 调用 `page_insert` 函数。
- 调用 `page_remove` 函数。
- 调用 `inverted_page_lookup` 函数。
- 修改某一个页表项。

我们保证上述前三个函数传入的 `pgdir` 均为 `inverted_page_lookup_test` 函数中第三行所定义的 `pgdir`。

我们保证所修改的页表项，一定是 `inverted_page_lookup_test` 函数中第三行所定义的 `pgdir` 对应**整个两级页表**中的页表项。

每调用一个函数或修改一个页表项，算一次操作，我们保证总操作数不超过 500。

运行 `make` 指令的最大时间为 10 秒，运行 Gxemul 模拟器的最大时间为 4 秒。

数据点说明

共有两组数据：

- 第一组数据为基本功能测试，实现完全正确才能通过评测，通过后可以获得 50 分。
  如果你的实现正确，评测机会返回 `Basic test passed!`，否则评测机会返回**第一个**错误之处：
  - 若函数 `page_insert` 的返回值有误，评测机会返回 `page_insert return value error!`。

    - 若 `vpn_buffer` 不满足严格递增，评测机会返回 `inverted_page_lookup vpn_buffer is not increasing!`。
    - 若存在一个**应当出现**的虚拟页号未出现在 `vpn_buffer` 中，评测机会返回 `inverted_page_lookup vpn_buffer is missing some vpn!`。
    - 若所有**应当出现**的虚拟页号均出现在 `vpn_buffer` 中，但函数 `inverted_page_lookup` 的返回值有误，评测机会返回 `inverted_page_lookup return value error!`。
    - 若你的程序出现其它错误，或未能在限定时间内执行全部程序，评测机会返回 `Other error!`。
- 第二组数据为 Hack 数据测试，实现完全正确才能通过评测，通过后可以获得 50 分。
  如果你的实现正确，评测机会返回 `Accepted!`，否则：
    - 若你的实现有误，评测机会返回 `Your implementation is wrong!`。
    - 若你的程序出现其它错误，或未能在限定时间内执行全部程序，评测机会返回 `Other error!`。
- **如果第一组数据未通过，则不会进行第二组数据的评测。**

测试样例

测试样例文件会上传至 MOOC 平台。

编写完成后，将 `init/init.c` 中的 `mips_init` 函数删除，并加入如下代码：

```
static void inverted_page_lookup_test(){
        struct Page *pp;
        page_alloc(&pp);
        Pde *pgdir = (Pde*)page2kva(pp);
        extern struct Page *pages;
        int a[10], len, i;
        page_insert(pgdir, pages + 2333, 0x23500000, 0);
        page_insert(pgdir, pages + 2333, 0x23400000, 0);
        page_insert(pgdir, pages + 2333, 0x23300000, 0);
        printf("%d\n", len = inverted_page_lookup(pgdir, pages + 2333, a));
        for(i = 0;i < len;i++) printf("%x\n", a[i]);
        page_remove(pgdir, 0x23400000);
        printf("%d\n", len = inverted_page_lookup(pgdir, pages + 2333, a));
        for(i = 0;i < len;i++) printf("%x\n", a[i]);
        page_insert(pgdir, pages + 2334, 0x23300000, 0);
        printf("%d\n", len = inverted_page_lookup(pgdir, pages + 2333, a));
        for(i = 0;i < len;i++) printf("%x\n", a[i]);
        printf("%d\n", len = inverted_page_lookup(pgdir, pages + 2334, a));
        for(i = 0;i < len;i++) printf("%x\n", a[i]);
}

void mips_init(){
        mips_detect_memory();
        mips_vm_init();
        page_init();

        inverted_page_lookup_test();

        *((volatile char*)(0xB0000010)) = 0;
}
```

运行如下指令：

```
make clean && make && /OSLAB/gxemul -E testmips -C R3000 -M 64 gxemul/vmlinux
```

正确输出如下：

```
3
23300
23400
23500
2
23300
23500
1
23500
1
23300
```

代码提交

```
git add .
git commit -m "xxxxx"
git push origin lab2-2-exam:lab2-2-exam
```

lab2-2 Extra

创建并切换分支

```
git checkout lab2
git add .
git commit -m "xxxxx"
git checkout -b lab2-2-Extra
```

若你想基于 lab2-2 exam 的代码进行开发，你可以使用：

```
git checkout lab2-2-exam
git add .
git commit -m "xxxxx"
git checkout -b lab2-2-Extra
```

题目描述

对于有多个内存条 (且访问每一个内存条的速度不一样) 的计算机系统来说，操作系统需要将频繁访问的内存块放到快内存条中，将鲜少访问的内存块放到慢内存条中，这种技术称为**内存迁移**。在内存迁移时，需要：

- 从目标内存条中找到空闲的物理页面 (即目标物理页面)。
- 将原物理页面中的内容复制到目标物理页面。
- 重新建立虚拟地址映射关系，即所有映射到原物理页面的虚拟页面，要重新映射到目标物理页面，并保证权限位不变。
- 若原物理页面已不被任何虚拟页面映射，则需要将其释放。

下面我们将模拟这样的内存迁移系统。

假设我们的 MOS 系统的 $64\ \mathrm{MB}$ 不是位于同一个内存条上的：低 $48\ \mathrm{MB}$ 位于慢内存条上，高 $16\ \mathrm{MB}$ 位于快内存条上。为减少同学们的实现难度与课程组的评测难度，我们做出如下约定：

- 低 $48\ \mathrm{MB}$ 对应的物理页面 (物理页号的范围是 $[0, 12287]$) 均使用原有代码的 `page_free_list` 空闲链表来管理，且原有的 `page_alloc` 均从 `page_free_list` 中获取页面结构体。

- 高 16 MB 对应的物理页面 (物理页号的范围是 $[12288, 16383]$) 均使用 `fast_page_free_list` 快页面空闲链表来管理，这个链表需要你在 `page_init` 中初始化。

请你对原有代码进行修改并实现页面迁移函数：

取消 `page_free_list` 的 `static` 属性

请你删除 `mm/pmap.c` 中全局变量 `page_free_list` 的 `static` 属性。这一步骤是为了方便课程组进行评测。

新建 `fast_page_free_list` 全局变量

请你在 `mm/pmap.c` 中参考 `page_free_list` 的定义，创建 `fast_page_free_list` 全局变量，注意不要保留其 `static` 属性。

修改原有物理内存管理初始化函数 `page_init`

- 函数原型不变：`void page_init(void)`
- 请你根据题目约定，将位于低 48 MB 的空闲物理页面对应的 `Page` 结构体放入 `page_free_list`，将位于高 16 MB 的空闲物理页面对应的 `Page` 结构体放入 `fast_page_free_list`。

修改原有物理页面释放函数 `page_free`

- 函数原型不变：`void page_free(struct Page *pp)`
- 调用此函数后，若 `pp` 指向的结构体中 `pp_ref` 已为 0，则根据 `pp` 对应的物理页面所在的内存条，放回到**对应**的空闲链表中。

新增内存迁移函数 `page_migrate`

- 函数原型：`struct Page* page_migrate(Pde *pgdir, struct Page *pp)`
- 调用此函数后，按顺序执行如下五个步骤：
  1. 确定目标内存条：
     - 若 `pp` 对应的物理页面位于慢内存条上，则目标内存条为快内存条。
     - 若 `pp` 对应的物理页面位于快内存条上，则目标内存条为慢内存条。
  2. 从目标内存条上选出一个空闲物理页面，假设 `Page` 结构体指针 `tp` 对应这个物理页面，将 `pp` 对应物理页面中的数据复制到 `tp` 对应的物理页面中。
  3. 找到 `pgdir` 对应的整个两级页表中所有映射到 `pp` 对应物理页面的虚拟页面，将他们映射到 `tp` 对应的物理页面上。在此过程中，需要保持原有权限位不变：

     > 若在迁移前，虚拟页面 `vp` 映射到原物理页面 `pp`，且映射的权限为 `PERM`，则将 `vp` 映射到目标物理页面 `tp` 后，映射的权限应仍为 `PERM`，不可增加或减少权限。

  4. 若 `pp` 对应的物理页面已不被任何虚拟页面映射，需要将其释放。

5. 返回 `tp` 。

注意事项

你需要先在 `include/pmap.h` 中加入如下函数定义：

```
struct Page* page_migrate(Pde *pgdir, struct Page *pp);
```

之后再在 `mm/pmap.c` 中实现这个函数。

若你需要修改 `Page` 结构体的定义，请你保证 `sizeof(struct Page)` 不超过 256。

请你保证执行 `page_init` 后，空闲链表 `page_free_list` 和快页面空闲链表 `fast_page_free_list` 中，地址越高的物理页面对应的结构体，越靠近链表头部。

评测逻辑

评测过程中，**我们会将所有的 `Makefile` 文件、`include.mk` 以及 `init/init.c` 替换为 lab2 初始配置**，接着将 `init/init.c` 中的 `mips_init` 函数改为如下形式：

```
void mips_init(){
        mips_detect_memory();
        mips_vm_init();
        page_init();

        page_migrate_test();

        *((volatile char*)(0xB0000010)) = 0;
}
```

最后的 `*((volatile char*)(0xB0000010)) = 0;` 会终止 Gxemul 模拟器的运行，避免占用评测资源。

`page_migrate_test()` 是在评测过程中新添加到 `init/init.c` 的函数，其形式为：

```
static void page_migrate_test(){
        struct Page *pp;
        page_alloc(&pp);
        Pde *pgdir = (Pde*)page2kva(pp);

        // test
}
```

`// test` 实际包含以下三种操作：

- 调用 `page_alloc` 函数。

- 调用 `page_insert` 函数。
- 调用 `page_migrate` 函数。

我们保证上述后两个传入的 `pgdir` 均为 `page_migrate_test` 函数中第三行所定义的 `pgdir`。

我们保证 `page_insert` 和 `page_migrate` 函数传入的页面结构体指针，一定是在此之前通过 `page_alloc` 或 `page_migrate` 所得到的。

每调用一个函数算一次操作，我们保证总操作数不超过 500。

运行 `make` 指令的最大时间为 10 秒，运行 Gxemul 模拟器的最大时间为 4 秒。

数据点说明

仅有一组数据，实现完全正确才能通过评测，通过后可以获得 100 分。

如果你的实现正确，评测机会返回 `Accepted!`，否则评测机会返回**第一个**错误之处：

- 若函数 `page_init` 未正确地将页面结构体放入 `page_free_list`，评测机会返回 `page_init page_free_list init error!`。
- 若函数 `page_init` 未正确地将页面结构体放入 `fast_page_free_list`，评测机会返回 `page_init fast_page_free_list init error!`。
- 若函数 `page_alloc` 的返回值有误，评测机会返回 `page_alloc return value error!`。
- 函数 `page_alloc` 会通过参数 `pp` 返回一个地址，若该地址有误，评测机会返回 `page_alloc pp error!`。
- 若函数 `page_insert` 的返回值有误，评测机会返回 `page_insert return value error!`。
- 若执行函数 `page_migrate` 后，页表项填写有误，评测机会返回 `page_migrate page table entry error!`。
- 若执行函数 `page_migrate` 后，物理页面内容复制有误，评测机会返回 `page_migrate physical page memory error!`。
- 若执行函数 `page_migrate` 后，未将正确的页面结构体从 `page_free_list` 中拿出，评测机会返回 `page_migrate page_free_list remove error!`。
- 若执行函数 `page_migrate` 后，未将正确的页面结构体从 `fast_page_free_list` 中拿出，评测机会返回 `page_migrate fast_page_free_list remove error!`。
- 若执行函数 `page_migrate` 后，未将正确的页面结构体放入 `page_free_list` 中，评测机会返回 `page_migrate page_free_list insert error!`。
- 若执行函数 `page_migrate` 后，未将正确的页面结构体放入 `fast_page_free_list` 中，评测机会返回 `page_migrate fast_page_free_list insert error!`。
- 若你的程序出现其它错误，或未能在限定时间内执行全部程序，评测机会返回 `Other error!`。

测试样例

测试样例文件会上传至 MOOC 平台。

编写完成后，将 `init/init.c` 中的 `mips_init` 函数删除，并加入如下代码：

```
static void page_migrate_test(){
        struct Page *pp;
        page_alloc(&pp);
        Pde *pgdir = (Pde*)page2kva(pp);
        page_alloc(&pp);
        page_insert(pgdir, pp, 0x23300000, 0);
        page_insert(pgdir, pp, 0x23400000, 0);
        page_insert(pgdir, pp, 0x23500000, 0);
        pp = page_migrate(pgdir, pp);
        printf("%d\n", page2ppn(pp));
        pp = page_migrate(pgdir, pp);
        printf("%d\n", page2ppn(pp));
}

void mips_init(){
        mips_detect_memory();
        mips_vm_init();
        page_init();

        page_migrate_test();

        *((volatile char*)(0xB0000010)) = 0;
}
```

运行如下指令：

```
make clean && make && /OSLAB/gxemul -E testmips -C R3000 -M 64 gxemul/vmlinux
```

正确输出如下：

```
16383
12286
```

代码提交

```
git add .
git commit -m "xxxxx"
git push origin lab2-2-Extra:lab2-2-Extra
```