

PA 3 实验报告

- Author: 张昱东
 - SID: 141120159
 - Email: 141120159@smail.nju.edu.cn
 - Time: 7月15日
-

一、实验进度

完成了所有实验内容：

- PA 3-1 Cache 的模拟 ✓
 - 单级 cache 的实现 (cache_read/write())
- PA 3-2 保护模式 ✓
 - 添加 CR0, GDTR, 段寄存器等
 - 虚拟地址向线性地址的转换 (segment_translate())
 - 段寄存器的填充 (load_sreg())
 - 特殊寄存器的 mov 指令, lgdt 指令, ljmp 指令
 - 主存分段式存储
- PA 3-3 虚拟地址转换 ✓
 - 添加 CR3
 - 线性地址向物理地址的转换 (page_translate())
 - 更新 loader() 的实现 (调用 mm_malloc())
 - 主存分页式存储 (和 PA 3-2 一起就是段页式存储)
 - TLB 的实现 (框架代码给出)

二、实验结果

由于整个 PA 3 的三个阶段的测试结果都一样，所以就放一张结果图

```

#ifndef __PA_CONFIG_H__
#define __PA_CONFIG_H__

// PA 2

// PA 3
#define CACHE_ENABLED
#define IA32_SEG           // protect mode enabled
#define IA32_PAGE          // virtual memory management is now complete
#define TLB_ENABLED

```

nemu trap output: [src/elf/elf.c,30,loader] {kernel} ELF loading from ram disk.
nemu: HIT GOOD TRAP at eip = 0x080481a8
NEMU2 terminated

Execute ./kernel/kernel.img ./ testcase/bin/sum
nemu trap output: [src/main.c,75,init_cond] {kernel} Hello, NEMU world!
nemu trap output: [src/elf/elf.c,30,loader] {kernel} ELF loading from ram disk.
nemu: HIT GOOD TRAP at eip = 0x080480dc
NEMU2 terminated

Execute ./kernel/kernel.img ./ testcase/bin/wanshu
nemu trap output: [src/main.c,75,init_cond] {kernel} Hello, NEMU world!
nemu trap output: [src/elf/elf.c,30,loader] {kernel} ELF loading from ram disk.
nemu: HIT GOOD TRAP at eip = 0x08048125
NEMU2 terminated

Execute ./kernel/kernel.img ./ testcase/bin/struct
nemu trap output: [src/main.c,75,init_cond] {kernel} Hello, NEMU world!
nemu trap output: [src/elf/elf.c,30,loader] {kernel} ELF loading from ram disk.
nemu: HIT GOOD TRAP at eip = 0x080481a0
NEMU2 terminated

Execute ./kernel/kernel.img ./ testcase/bin/string
nemu trap output: [src/main.c,75,init_cond] {kernel} Hello, NEMU world!
nemu trap output: [src/elf/elf.c,30,loader] {kernel} ELF loading from ram disk.
nemu: HIT GOOD TRAP at eip = 0x0804820a
NEMU2 terminated

Execute ./kernel/kernel.img ./ testcase/bin/hello-str
nemu trap output: [src/main.c,75,init_cond] {kernel} Hello, NEMU world!
nemu trap output: [src/elf/elf.c,30,loader] {kernel} ELF loading from ram disk.
nemu: HIT GOOD TRAP at eip = 0x080481a5
NEMU2 terminated

Execute ./kernel/kernel.img ./ testcase/bin/test-float
nemu trap output: [src/main.c,75,init_cond] {kernel} Hello, NEMU world!
nemu trap output: [src/elf/elf.c,30,loader] {kernel} ELF loading from ram disk.
nemu: HIT BAD TRAP at eip = 0x0804815a
NEMU2 terminated

三、必答题

PA 3-2

1. NEMU 在什么时候进入了保护模式?

- nemu 刚启动时进入的是实模式。在实模式下，nemu 需要初始化段表 GDT 和描述符表寄存器 GDTR。在初始化完成后，nemu 通过把 CR0 中的 PE 位置为 1，从而进入保护模式。
- 在具体实现中，对应的就是这段汇编代码

```
start:  
ifdef IA32_INTR  
    cli  
endif  
    lgdt    va_to_pa(gdtdesc) # See i386 manual for more information  
    movl    %cr0, %eax        # %CR0 |= PROTECT_ENABLE_BIT  
    orl    $0x1, %eax  
    movl    %eax, %cr0
```

- 通过 movl %eax, %cr0 来通知进入保护模式

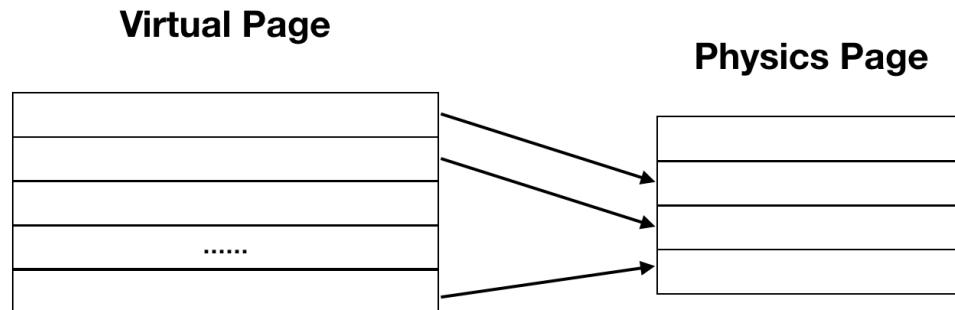
2. 在 GDTR 中保存的段表首地址是虚拟地址、线性地址、还是物理地址？为什么？

- 是线性地址 (如果分页机制没打开的话，其实就是物理地址)
- 由于 GDTR 的填充是在保护模式开始之前的，所以在保护模式开始之前分段机制尚未开启，所以 GDTR 中填充的地址都是线性地址；
- 分段机制开始之后的指令中用到的地址才是虚拟地址，需要通过地址转换来变成线性地址。

PA 3-3

3. Kernel 的虚拟页和物理页的映射关系是什么？请画图说明；

- 映射关系是全相连映射，因为缺页的开销很大，所以要提高命中率
- 如图

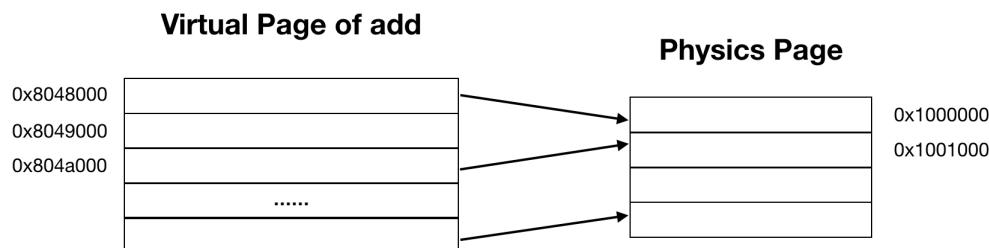


4. 以某一个测试用例为例，画图说明用户进程的虚拟页和物理页间映射关系又是怎样的？Kernel 映射为哪一段？你可以在 `loader()` 中通过 `Log()` 输出 `mm_malloc` 的结果来查看映射关系，并结合 `init_mm()` 中的代码绘出内核映射关系；

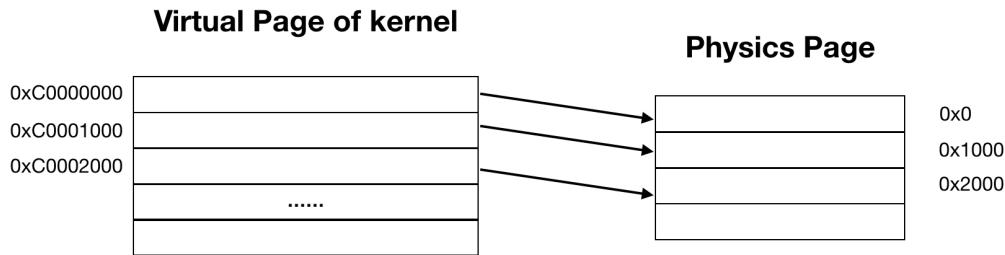
- 以测试样例 add 为例，通过 Log 输出装载结果如下

```
Execute ./kernel/kernel.img ./testcase/bin/add
nemu trap output: [src/main.c,75,init_cond] {kernel} Hello, NEMU world!
nemu trap output: [src/elf/elf.c,30,loader] {kernel} ELF loading from ram disk.
nemu trap output: [src/elf/elf.c,38,loader] {kernel}
PT Load:
nemu trap output: [src/elf/elf.c,53,loader] {kernel} vaddr = 0x8048000
nemu trap output: [src/elf/elf.c,54,loader] {kernel} paddr = 0x1000000
nemu trap output: [src/elf/elf.c,55,loader] {kernel} memsz = 0x1e8
nemu trap output: [src/elf/elf.c,38,loader] {kernel}
PT Load:
nemu trap output: [src/elf/elf.c,53,loader] {kernel} vaddr = 0x804a000
nemu trap output: [src/elf/elf.c,54,loader] {kernel} paddr = 0x1001000
nemu trap output: [src/elf/elf.c,55,loader] {kernel} memsz = 0x140
nemu: HIT GOOD TRAP at eip = 0x08048151
NEMU2 terminated
```

- 在 add 样例中，虚拟页和物理页的映射关系如图（全相连映射）



- 从 `init_mm()` 的代码中可以看出虚拟内存空间中 0xC0000000 以上的 kernel 代码被映射到了 0x0 开始的物理内存空间



5. “在 Kernel 完成页表初始化前，程序无法访问全局变量”这一表述是否正确？在 `init_page()` 里面我们对全局变量进行了怎样的处理？
- 正确
 - 因为在 kernel 完成页表初始化开启分页模式之前，我们访问内存单元使用的地址都是物理地址（段基址被设置成 0）。而我们在编译 kernel 时，全局变量的地址都是高地址，不能直接使用。
 - 在 `init_page()` 中对全局变量的使用都是通过宏 `va_to_pa`，通过减去 `KOFFSET`，使全局变量的地址变成一个低地址，从而可以对应到内存上而被使用。

四、个人收获

- PA 3 的实验内容和教材相关内容结合比较紧密，如果对教材里的内容（各种存储机制）理解透了，那么对于 PA 的思路和要完成的内容都会有一个比较清晰的认识。难点就剩如何用代码来实现这些机制了。
- 相比于之前的阶段，PA 3 关于具体实现的提示就简略了很多，刚拿到也比较难下手。而且由于改用 testkernel 后就不方便进入 nemu 的交互状态，比较难 debug。虽然代码量不大，但是时间都主要用在了思考实现方式和 debug 上了。
- 在实现 cache 的时候还遇到了一个以假乱真的 bug

```

Execute ./ testcase/bin/struct.img ./ testcase/bin/struct
Parsing error! Wrong paddr.
EIP = 0x30200
cur_addr = 0xffffffff, tag = 0xffff, set = 0x7f, block = 0x3f
Parsing error! Wrong paddr.
EIP = 0x30265
cur_addr = 0xffffffff, tag = 0xffff, set = 0x7f, block = 0x3f
nemu: HIT GOOD TRAP at eip = 0x000302b6
NEMU2 terminated

Execute ./ testcase/bin/string.img ./ testcase/bin/string
Parsing error! Wrong paddr.
EIP = 0x30200
cur_addr = 0xffffffff, tag = 0xffff, set = 0x7f, block = 0x3f
Parsing error! Wrong paddr.
EIP = 0x30265
cur_addr = 0xffffffff, tag = 0xffff, set = 0x7f, block = 0x3f
nemu: HIT GOOD TRAP at eip = 0x000302b6
NEMU2 terminated

Execute ./ testcase/bin/hello-str.img ./ testcase/bin/hello-str
Parsing error! Wrong paddr.
EIP = 0x30200
cur_addr = 0xffffffff, tag = 0xffff, set = 0x7f, block = 0x3f
Parsing error! Wrong paddr.
EIP = 0x30265
cur_addr = 0xffffffff, tag = 0xffff, set = 0x7f, block = 0x3f
nemu: HIT GOOD TRAP at eip = 0x000302b6
NEMU2 terminated

Execute ./ testcase/bin/test-float.img ./ testcase/bin/test-float
Parsing error! Wrong paddr.
EIP = 0x30200
cur_addr = 0xffffffff, tag = 0xffff, set = 0x7f, block = 0x3f
Parsing error! Wrong paddr.
EIP = 0x30265
cur_addr = 0xffffffff, tag = 0xffff, set = 0x7f, block = 0x3f
nemu: HIT GOOD TRAP at eip = 0x000302b6
NEMU2 terminated

```

- o 看似通过了样例测试，但是却发现即使是 test-float 都 hit good trap 了！
- o 经过仔细分析后发现，所有样例 hit good trap 时的 eip 都一样，考虑到所有样例的代码在内存中的起始位置 (eip) 都一样，因此执行所有样例时，实际上在 cache 中读到的都是第一个样例的代码，所以每次都在同样的地方 hit good trap
- o 检查代码后发现，确实忘记在 restart() 里刷新 cache，所以还残留了上一个进程装载的内容，在本次执行中被访问到了
- 还有一个 bug 也是在实现 cache 时遇到的，就是在解析物理地址时，发现样例给

cache_read 传了一个非法的地址，即超过了主存大小

- 使用 cache_read() 时遇到这样的报错

```
Execute ./kernel/kernel.img ./ testcase/bin/mov
nemu trap output: [src/main.c,75,init_cond] {kernel} Hello, NEMU world!
nemu trap output: [src/elf/elf.c,29,loader] {kernel} ELF loading from ram disk.
Parsing error! Wrong paddr.
EIP = 0x60200
cur_addr = 0xffffffff, tag = 0x7fffff, set = 0x7f, block = 0x3f
Parsing error! Wrong paddr.
EIP = 0x60265
cur_addr = 0xffffffff, tag = 0x7fffff, set = 0x7f, block = 0x3f
nemu: HIT GOOD TRAP at eip = 0x000602b6
NEMU2 terminated

Execute ./kernel/kernel.img ./ testcase/bin/mov-cmp
nemu trap output: [src/main.c,75,init_cond] {kernel} Hello, NEMU world!
nemu trap output: [src/elf/elf.c,29,loader] {kernel} ELF loading from ram disk.
Parsing error! Wrong paddr.
EIP = 0x602c8
cur_addr = 0xffffffff, tag = 0x7fffff, set = 0x7f, block = 0x3f
Parsing error! Wrong paddr.
EIP = 0x6032d
cur_addr = 0xffffffff, tag = 0x7fffff, set = 0x7f, block = 0x3f
nemu: HIT GOOD TRAP at eip = 0x0006037e
NEMU2 terminated
```

- 但是如果用 hw_mem_read 就没有

```
Execute ./kernel/kernel.img ./ testcase/bin/mov
nemu trap output: [src/main.c,75,init_cond] {kernel} Hello, NEMU world!
nemu trap output: [src/elf/elf.c,29,loader] {kernel} ELF loading from ram disk.
nemu: HIT GOOD TRAP at eip = 0x000602b6
NEMU2 terminated

Execute ./kernel/kernel.img ./ testcase/bin/mov-cmp
nemu trap output: [src/main.c,75,init_cond] {kernel} Hello, NEMU world!
nemu trap output: [src/elf/elf.c,29,loader] {kernel} ELF loading from ram disk.
nemu: HIT GOOD TRAP at eip = 0x0006037e
NEMU2 terminated
```

- 分析过程：PA 里模拟的内存大小是128M，所以主存地址位数应该是27位，那么tag、组索引、块内索引的位数就是 14+7+6，那么从一个正常的地址中解析出来的 tag 就应该要小于 2^{14} ，即小于 0x4000，但是这个命令中的 tag 显然大于了 0x4000（这个内存地址也超过内存大小了）
- 通过“插桩 debug 法”解决了问题，出错的地方是 For 循环的终止条件问题，因为 $-1 < 0$ ，本来应该终止，结果由于是 paddr（本质是 uint32_t 类型），所以比较结果是 -1 更大，所以循环未终止，导致 -1（即 0xffffffff）也被当地址解析，导致出错
- 在做 PA 3 时虽然遇到的 bug 多多，而且对于实现方法有时也很迷茫，但是总体压力还是比 PA 2 轻松一点的。通过 PA 3 的实践，也对 cache 机制、分段式存储、分页式存储的机制和原理有了更深的理解。