南京大学

PA-3-2：保护模式

课程名称：计算机系统基础

姓名：孙文博

学号：201830210

邮箱：201830210@smail.nju.edu.cn

实验时间：2022.6.1 - 2022.6.13

一、 实验进度

1. 了解保护模式，逻辑地址，段寄存器等背景知识；
2. 修改 include/config.h 头文件，为 NEMU 开启保护模式；
3. 在 nemu/include/cpu/reg.h 头文件中，为 CPU_STATE 结构添加上必要的器件模拟，其中包括 GDTR、CR0 和长度为 6 的段寄存器数组 SegReg[]：

```
typedef struct {
    uint32_t limit :16;
    uint32_t base :32;
}GDTR;

typedef union {
    struct {
        uint32_t pe :1;
        uint32_t mp :1;
        uint32_t em :1;
        uint32_t ts :1;
        uint32_t et :1;
        uint32_t reserve :26;
        uint32_t pg :1;
    };
    uint32_t val;
}CR0;

typedef struct {
    // the 16-bit visible part, i.e., the selector
    union {
        uint16_t val;
        struct {
            uint32_t rpl :2;
            uint32_t ti :1;
            uint32_t index :13;
        };
    };

    // the invisible part, i.e., cache part
    struct {
        uint32_t base;
        uint32_t limit;
        uint32_t type :5;
        uint32_t privilege_level :2;
        uint32_t soft_use :1;
    };
}SegReg;
```

4. 实现 lgdt 指令及 mov，jmp 指令中有关分段的部分；
5. 补全 memory/mmu/segment.c 文件中 segment_translate()、

loag_sreg()函数：

```
uint32_t segment_translate(uint32_t offset, uint8_t sreg)
{
    /* TODO: perform segment translation from virtual address to linear address
     * by reading the invisible part of the segment register 'sreg'
     */
    uint32_t addr = cpu.segReg[sreg].base + offset;
    return addr;
}

// load the invisible part of a segment register
void load_sreg(uint8_t sreg)
{
    /* TODO: load the invisibile part of the segment register 'sreg' by reading the GDT.
     * The visible part of 'sreg' should be assigned by mov or ljmp already.
     */
    SegDesc new_seg;
    memcpy(&new_seg, hw_mem+cpu.gdtr.base+cpu.segReg[sreg].index*8, sizeof(new_seg));

    uint32_t base = (new_seg.base_31_24 << 24) + (new_seg.base_23_16 << 16) + new_seg.base_15_0;
    uint32_t limit = (new_seg.limit_19_16 << 16) + new_seg.limit_15_0;

    assert(base == 0);
    assert(limit <= 0xffffffff);
    assert(new_seg.granularity == 1);

    cpu.segReg[sreg].base = base;
    cpu.segReg[sreg].limit = limit;
    cpu.segReg[sreg].type = new_seg.type;
    cpu.segReg[sreg].privilege_level = new_seg.privilege_level;
    cpu.segReg[sreg].soft_use = new_seg.soft_use;
}

```

6. 在 vaddr_read() 和 vaddr_write() 函数中添加保护模式下的虚拟地址向线性地址转换的过程：

```
uint32_t vaddr_read(vaddr_t vaddr, uint8_t sreg, size_t len)
{
    assert(len == 1 || len == 2 || len == 4);
#ifdef IA32_SEG
    uint32_t laddr=vaddr;
    if(cpu.cr0.pe)
    {
        laddr=segment_translate(vaddr,sreg);
    }
    return laddr_read(laddr,len);

#else
    return laddr_read(vaddr, len);
#endif
}

void vaddr_write(vaddr_t vaddr, uint8_t sreg, size_t len, uint32_t data)
{
    assert(len == 1 || len == 2 || len == 4);
#ifdef IA32_SEG
    if(cpu.cr0.pe)
    {
        vaddr=segment_translate(vaddr,sreg);
        laddr_write(vaddr,len,data);
    }
    else
        laddr_write(vaddr,len,data);
#else
    laddr_write(vaddr, len, data);
#endif
}
```

7. 执行 make test_pa-3-2 命令并通过各测试用例：

```
NEMU2 terminated
./nemu/nemu --autorun --testcase struct --kernel
NEMU load and execute img: ./kernel/kernel.img  elf: ./testcase/bin/struct
nemu trap output: [src/main.c,82,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 = 0x0010010c
NEMU2 terminated
./nemu/nemu --autorun --testcase string --kernel
NEMU load and execute img: ./kernel/kernel.img  elf: ./testcase/bin/string
nemu trap output: [src/main.c,82,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 = 0x0010016a
NEMU2 terminated
./nemu/nemu --autorun --testcase hello-str  --kernel
NEMU load and execute img: ./kernel/kernel.img  elf: ./testcase/bin/hello-str
nemu trap output: [src/main.c,82,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 = 0x00100105
NEMU2 terminated
./nemu/nemu --autorun --testcase test-float --kernel
NEMU load and execute img: ./kernel/kernel.img  elf: ./testcase/bin/test-float
nemu trap output: [src/main.c,82,init_cond] {kernel} Hello, NEMU world!
nemu trap output: [src/elf/elf.c,29,loader] {kernel} ELF loading from ram disk.
nemu: HIT BAD TRAP at eip = 0x001000c8
NEMU2 terminated
make-[1]: Leaving directory '/home/pa201830210/pa_nju'
pa201830210@edb32e250119:~/pa_nju$
```

二、 思考题

1. NEMU 在什么时候进入了保护模式?

答: nemu 的头文件 config.h 中定义了 IA32_SEG 之后，可以通过判断 CR0 的 PE 是否为 1 来判断 NEMU 是否开启保护模式

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?

答: 是线性地址；因为在分段机制中，需要完成从虚拟地址到线性地址的转换，给出的 48 位地址中，有 32 位的偏移量，这个偏移量显然

是在线性地址中的偏移量，所以 GDTR 保存的段表首地址显然也是线性地址。

三、 总结与反思

本阶段的内容与书本知识联系比较紧密，同时分段和分页知识也是整个 PA 及 ICS 中的重点部分，有一定的难度与理解上的挑战。虽然过程中出现了一些 bug，一度导致之前的 3-1 和 2-2 测试都出现问题，但好在仔细分析报错信息之后，成功修复。希望下阶段实验顺利！