软件综合实验之操作系统

进入保护模式

陈香兰

中国科学技术大学计算机学院

July 1, 2016

提纲

实验准备

- 实验环境准备
  - 编译工具链：gcc、ld
  - 模拟环境：qemu-system-i386
- 基础知识和技能准备
  - BIOS中断和键盘输入
  - i386保护模式
  - 使用hexdump查看bin文件和img文件中的内容
    （对照：尝试查看你磁盘的MBR的内容）
  - 使用objdump反汇编目标文件“xxx.o”和elf文件
    （对照：尝试反汇编“xxx.bin”文件）

BIOS中断和键盘输入

- BIOS中断参考：
  https://en.wikipedia.org/wiki/BIOS_interrupt_call#Interrupt_table

1. 显示相关：int 0x10
   - 作业1：请将示意代码中的字符输出改成使用BIOS中断的方式
2. 键盘相关：int 0x16

AH	Description
00h	Read Character

i386的分段机制

- I386体系结构采用分段机制
  - 逻辑地址=段：段内偏移
- 使用16位段寄存器来指明当前所使用的段
  - 有六个：cs，ss，ds，es，fs和gs
  - CPU规定了3个寄存器的专门的用途
    - cs 代码段寄存器，指向存放程序指令的段
    - ss 堆栈段寄存器，指向存放当前堆栈的段
    - ds 数据段寄存器，指向存放数据的段

I386的地址转换模式：实模式和保护模式

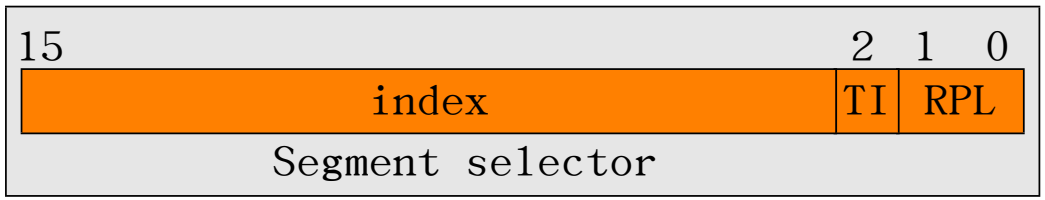
- 实模式（20位）
  - 16位段寄存器只记录段基址的高16位，
    因此段基址必须4位对齐（末4位为0）
  - 不采用虚拟地址空间，直接采用物理地址空间
  - 物理地址=段寄存器值*16+段内偏移
- 保护模式（32位）
  - 16位段寄存器无法直接记录段的信息，
    因此需要与全局描述符表GDT配合使用
  - GDT中记录了每个段的信息（段描述符），
    段寄存器只需记录段在GDT中的序号

GDT表

- One 8-bytes segment descriptor for each segment
- GDT is stored in memory
- Registers GDTR stores the base address of a GDT
  - The GDTR is 48 bits long
  - The lower 16 bits tell the size of the GDT, and the upper 32 bits tell the location of the GDT in memory.
  - instruction: lgdt src

GDT表

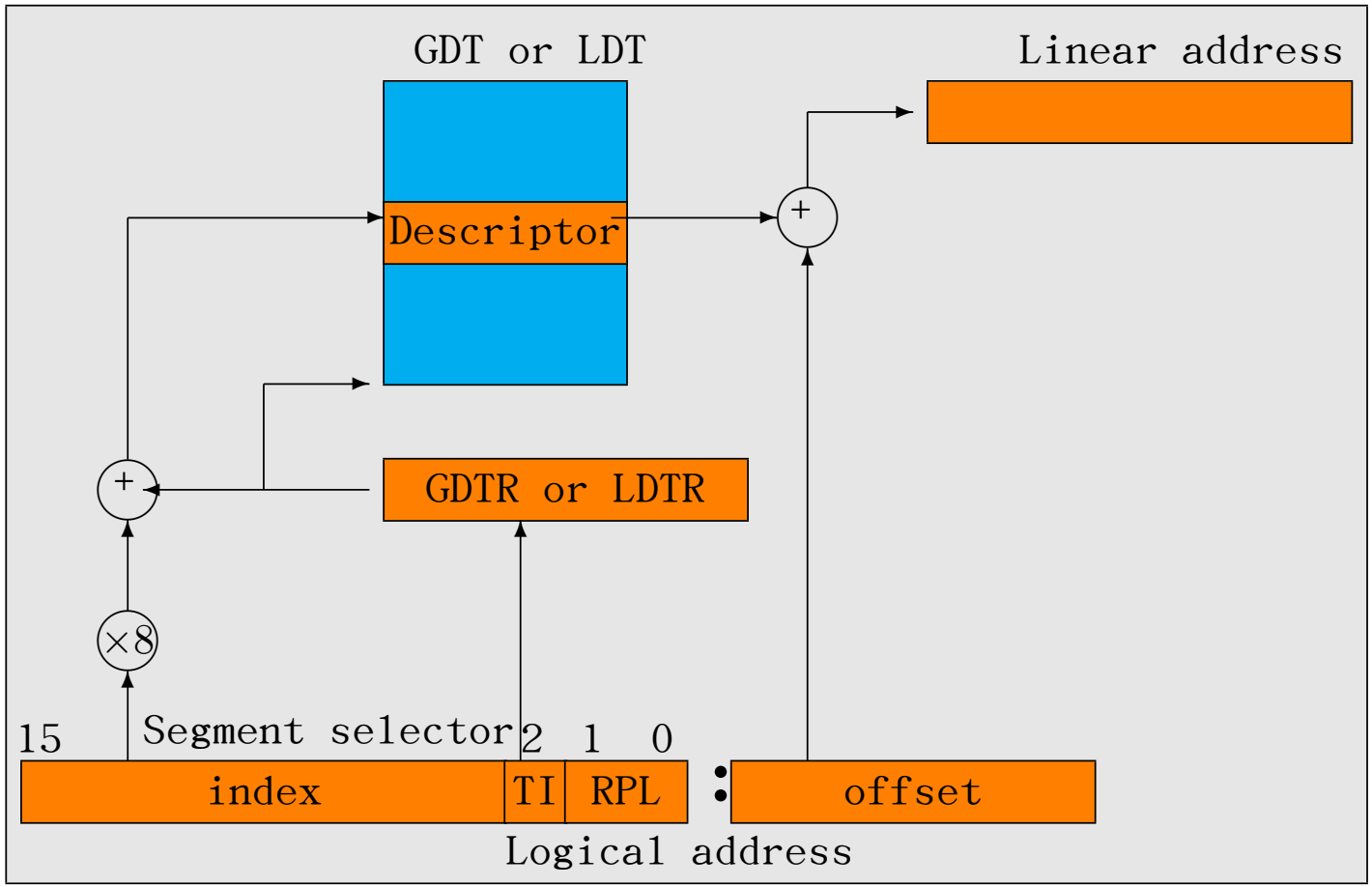
- Segment selector (段选择子): The value in segment register, 16-bits



1. Index: 13 bits, the index of corresponding segment descriptor in GDT
2. Table Indicator, TI-bit: 1 bit, GDT? LDT?
3. Request privilege level, RPL-bits: 2 bits

GDT表

- Linear address = segment base + offset



GDT表

- Types of segment descriptors:
  1. Data Segment Descriptor (数据段描述符): for data/stack segments
  2. Code Segment Descriptor (代码段描述符): for code segments
  3. Task State Segment Descriptor (任务状态段描述符)
  4. LDT Descriptor (LDT描述符)
  5. System Segment Descriptor (系统段描述符)

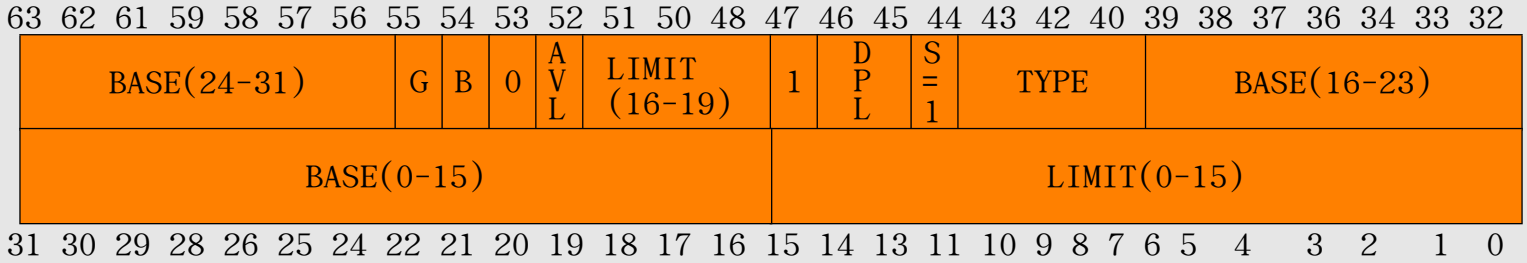
GDT表

- Contents of segment descriptors:
  - Base (32-bits): Segment start address in physical memory
  - Limit (20-bits): for segment length
  - G-bit (1-bit): the unit of segment length (0: 1= 1B; 1: 1=4KB)
  - S-bit (1-bit): system segment (0) or not (1)
  - Type (4-bits): for code/data/tss/ldt/etc BIOS中断和键盘输入
  - DPL-bits (2-bits): descriptor privilege level of the segment (00b~11b)
  - Segment present flag (1-bit): present (1) or not (0)
  - ...

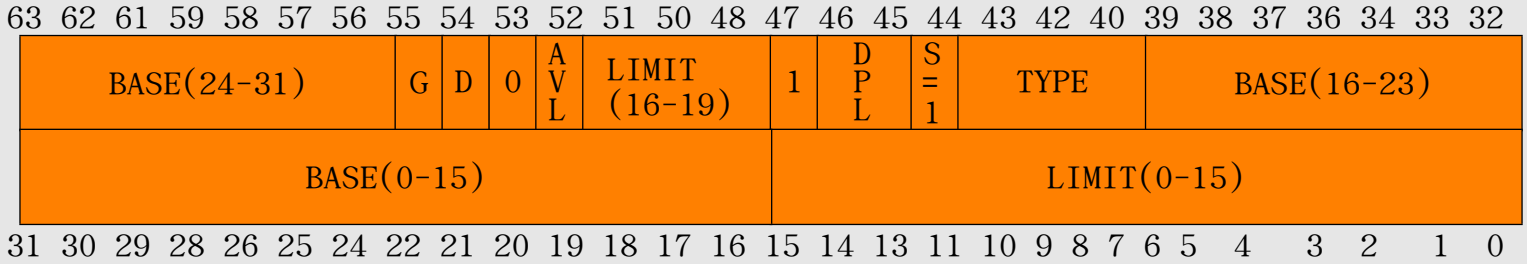
GDT表

- Contents of segment descriptors:

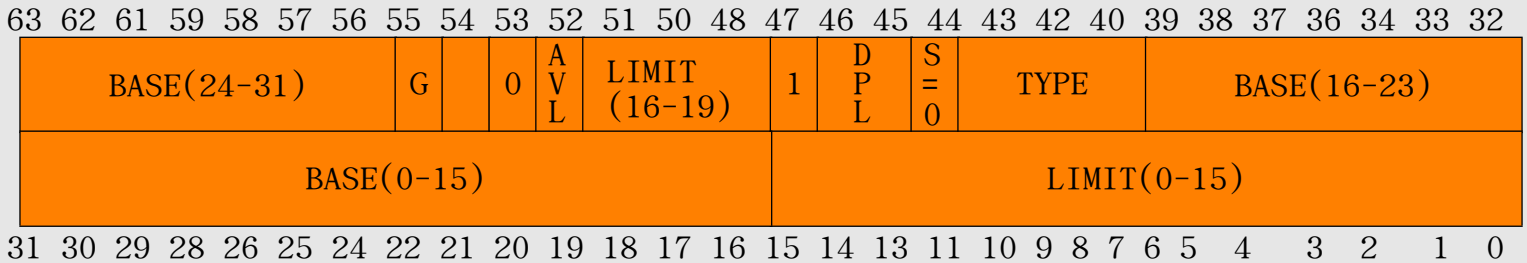
Data Segment Descriptor



Code Segment Descriptor

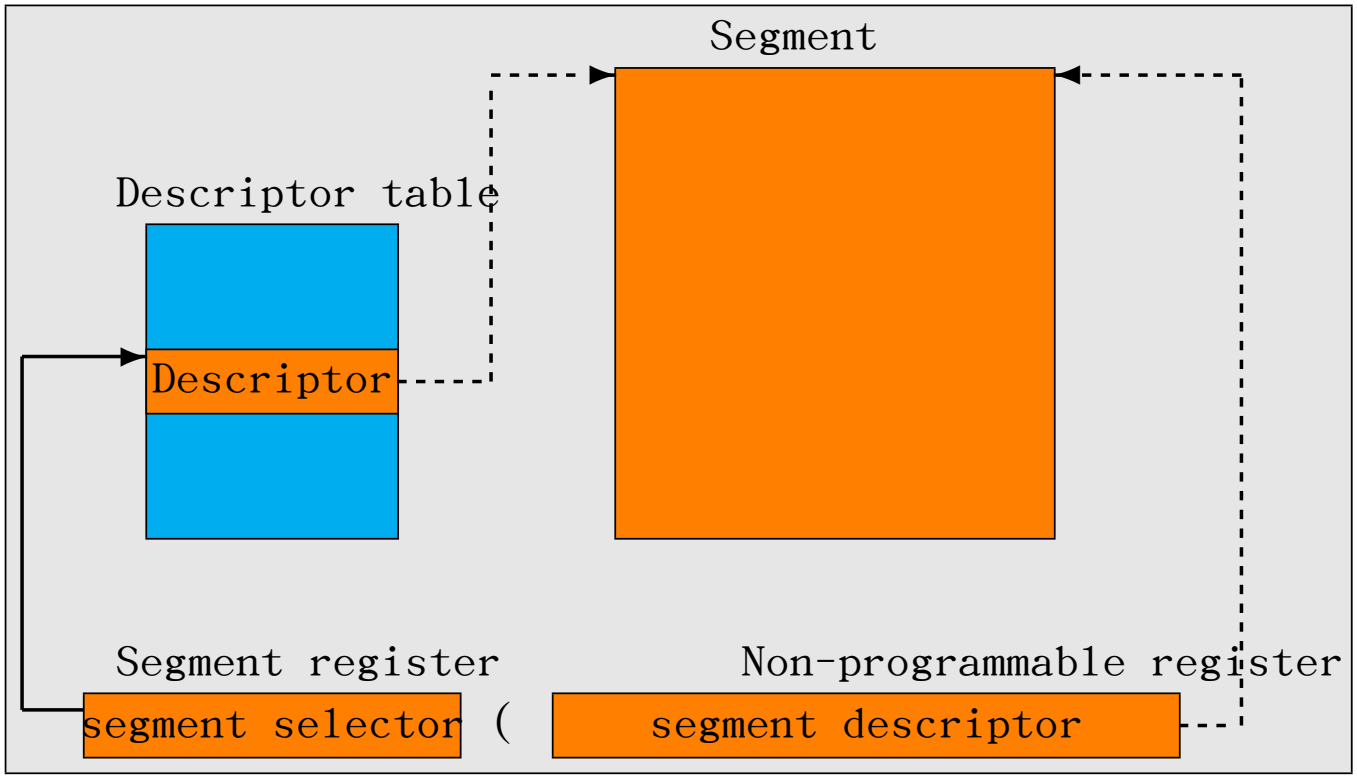


System Segment Descriptor



GDT表

- Selector and the quick access to descriptor



打开保护模式开关

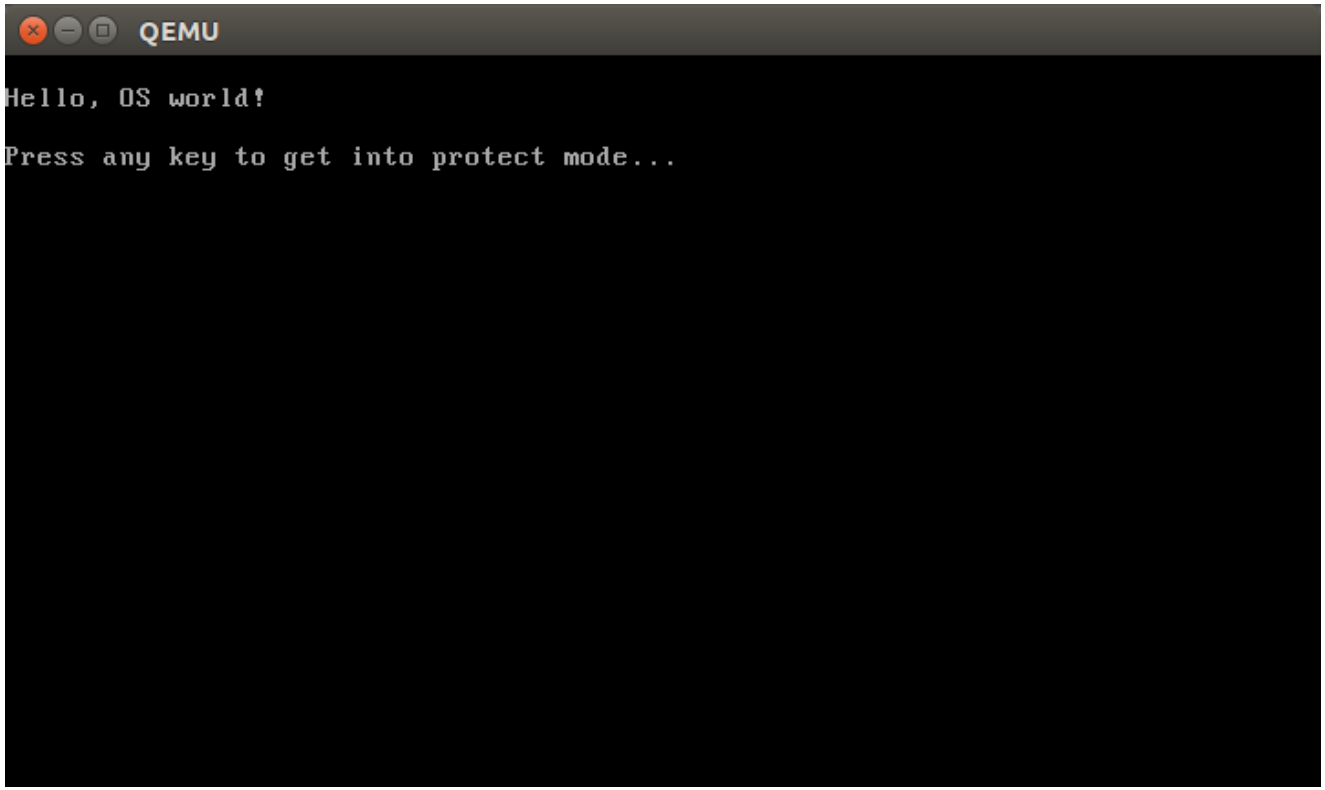
- 控制寄存器CR0的最后一位是保护模式开关

从实模式进入保护模式

1. 阅读源代码
   - start16_rm2pm.S
   - start16_rm2pm.ld
2. 编译链接并制作成二进制映像
   - gcc -c -m32 start16_rm2pm.S -o start16_rm2pm.o
   - ld -T start16_rm2pm.ld start16_rm2pm.o -o start16_rm2pm.elf
   - objcopy -O binary start16_rm2pm.elf start16_rm2pm.bin
3. 制作启动软盘
   - dd if=/dev/zero of=a_start16_rm2pm.img bs=512 count=2880
   - sudo losetup /dev/loop4 a_start16_rm2pm.img
   - sudo dd if=start16_rm2pm.bin of=/dev/loop4 bs=512 count=1

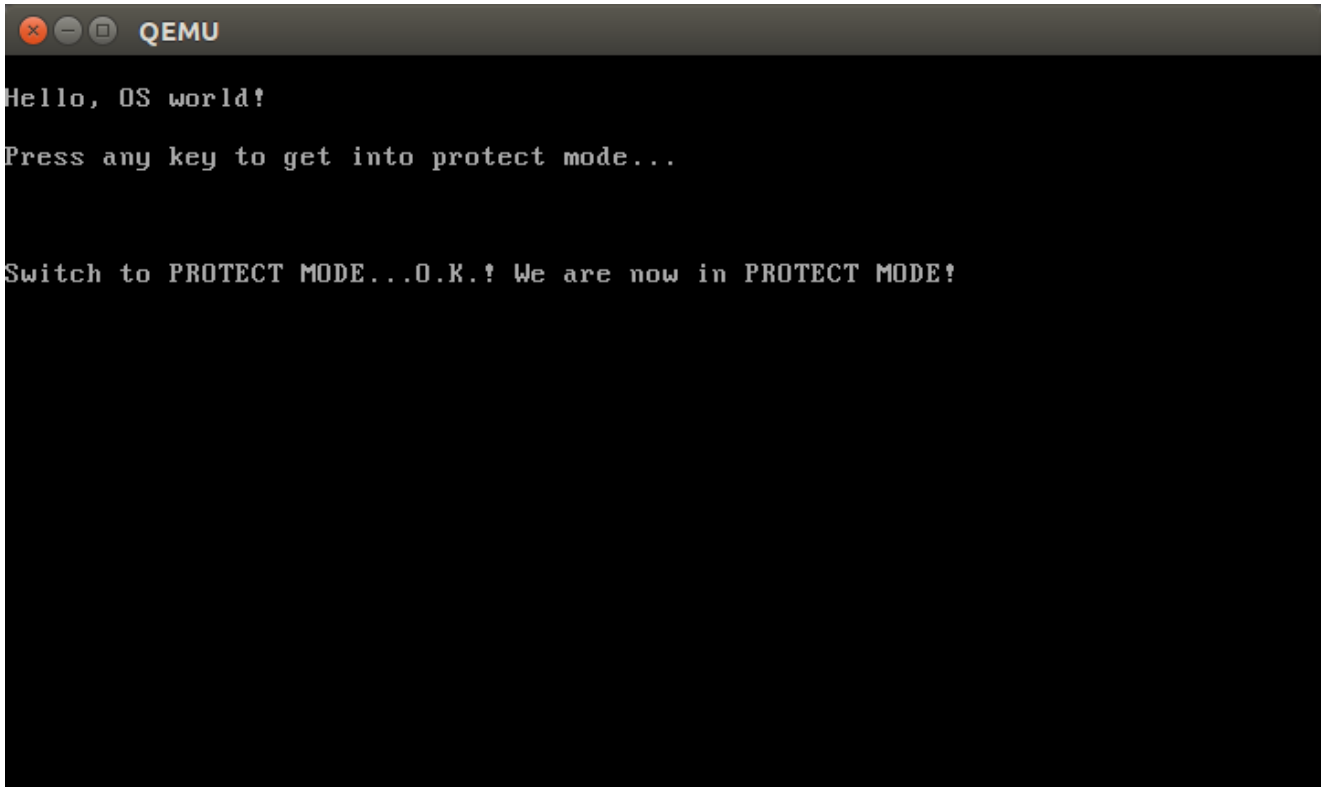
从实模式进入保护模式

4. 在qemu上启动
   - qemu-system-i386 -fa a_start16_rm2pm.img



从实模式进入保护模式

4. 在qemu上启动
   - qemu-system-i386 -fa a_start16_rm2pm.img



作业：请在”O.K.!”后添加回车换行，
另起一行输出”We are now in PROTECT MODE!”

使用hexdump

- 查看bin文件
  - hexdump -C output/start16_rm2pm.bin
- 查看img文件
  - hexdump -C output/a_start16_rm2pm.img
- 查看我磁盘MBR的内容
  - sudo dd if=/dev/sda of=sda_mbr.bin bs=512 count=1
    (其中，/dev/sda是你的磁盘，若你有多个磁盘，可以选择不同的磁盘查
    (还可能是/dev/hd*)
  - hexdump -C sda_mbr.bin

小结

谢谢！