问题清单

在整个实验的过程中，无论是编程还是查资料，请各位同学注意思考以下问题，助教检查时会从中随机抽取数个题目进行提问，根据现场作答给出分数。请注意，我们鼓励自己思考和动手实验，如果能够提供自己的思考结果并辅助以相应的实验结果进行说明，在分数评定上会酌情考虑。

2.1 PPT相关内容

1. **什么是实模式，什么是保护模式?**
    - 实模式：用基地址加偏移量就可以直接拿到物理地址的模式。（非常不安全）
    - 保护模式：不能直接拿物理地址的模式。（需要进行地址转换，从80386开始，是现代操作系统的主要模式）
2. **什么是选择子?**
    - 作用是作为索引选择描述符表中的描述符项。
    - 选择子共16位，放在段选择寄存器里
    - 低2位表示请求特权级RPL（与用户态和内核态相关）
    - 第3位表示选择GDT方式还是LDT方式
    - 高13位表示在描述符表中的偏移（故描述符的项数最多是2的13次方）
3. **什么是描述符?**
    - 存放在GDT或LDT中，每一个描述符定义了一个段。
    - 包含段基址（32位），段界限（20位）以及各种描述符属性，三者交错存放。

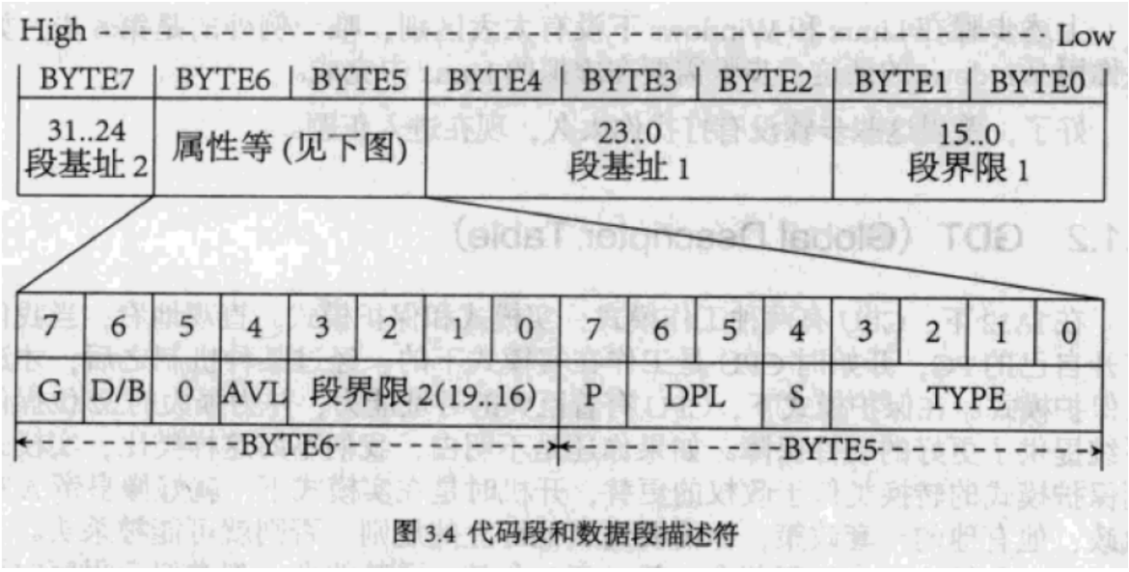


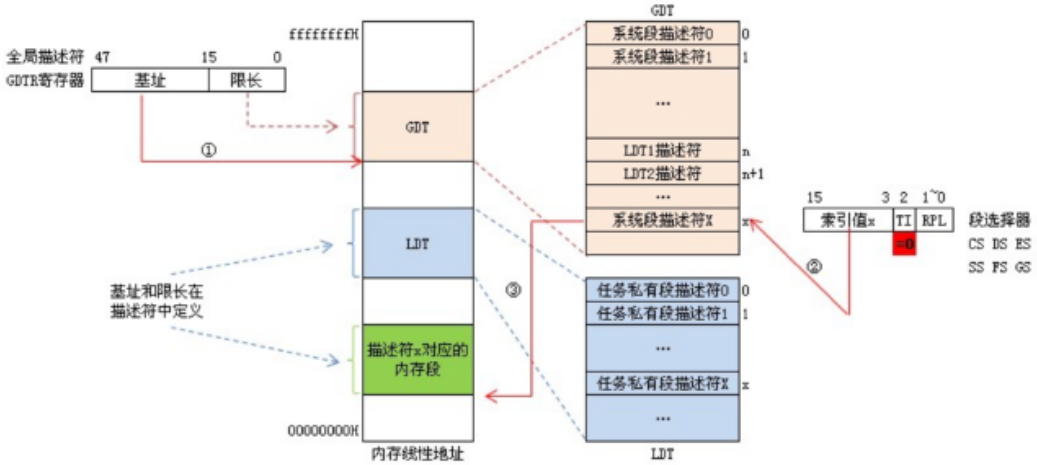
图 3.4 代码段和数据段描述符

    - 描述符属性
        - P位：存在（Present）位。P=1表示段在内存中存在；P=0表示段在内存中不存在。
        - DPL：描述符特权级。特权级可以是0，1，2，3. 数字越小特权级越大。
        - S位：指明描述符是数据段/代码段描述符（S=1）还是系统段/门描述符（S=0）
        - TYPE：数据段和代码段描述符的读写权限等，或系统段和门描述符的门等。（OrgP36）
        - G：段界限粒度位。G=0时段界限粒度为字节，G=1时段界限粒度为4KB.
        - D/B：根据描述符种类不同有不同用法。

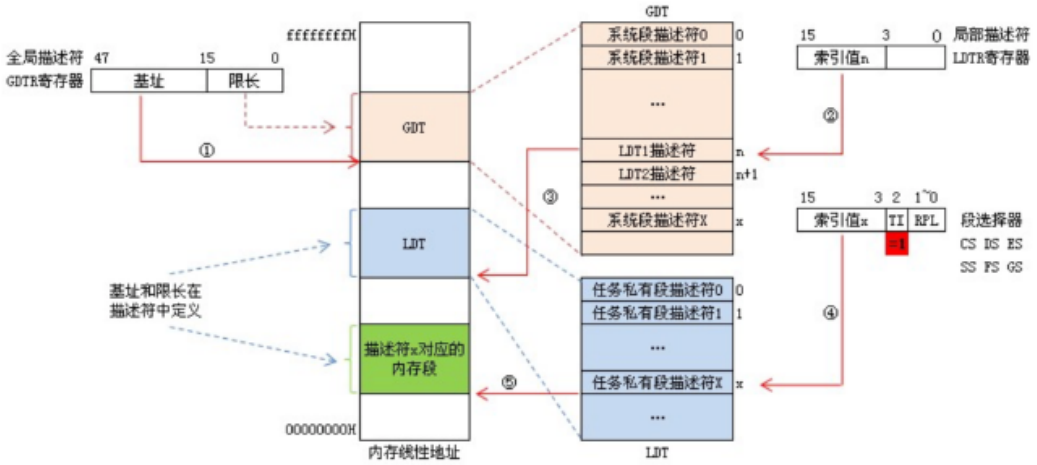
- 在可执行代码段描述符中，这一位叫做 D 位。D=1 时，在默认情况下指令使用 32 位地址及 32 位或 8 位操作数；D=0 时，在默认情况下使用 16 位地址及 16 位或 8 位操作数。
- 在向下扩展数据段的描述符中，这一位叫做 B 位。B=1 时，段的上部界限为 4GB；B=0 时，段的上部界限为 64KB。
- 在描述堆栈段（由ss寄存器指向的段）的描述符中，这一位叫做 B 位。B=1 时，隐式的堆栈访问指令（如push、pop和call）使用 32 位堆栈指针寄存器esp；D=0 时，隐式的堆栈访问指令（如push、pop和call）使用 16 位堆栈指针寄存器sp。

    - AVL：保留位。

4. **什么是GDT，什么是LDT?**
    - GDT：全局描述符表，是全局唯一的。存放一些公用的描述符、和包含各进程局部描述符表首地址的描述符。
    - LDT：局部描述符表，每个进程都可以有一个。存放本进程内使用的描述符。
5. **请分别说明GDTR和LDTR的结构。**
    - GDTR：48位寄存器，高32位放置GDT首地址，低16位放置GDT限长（限长决定了可寻址的大小，注意低16位放的不是选择子）
    - LDTR：16位寄存器，放置一个特殊的选择子，用于查找当前进程的LDT首地址。
6. **请说明GDT直接查找物理地址的具体步骤。**
    1. 给出段选择子（放在段选择寄存器里）+偏移量
    2. 若选择了GDT方式，则从GDTR的高32位获取GDT首地址，用段选择子中的高13位做偏移，拿到GDT中的描述符。
    3. 如果合法且有权限，用描述符中的段首地址加上（1）中的偏移量找到物理地址。寻址结束。



7. **请说明通过LDT查找物理地址的具体步骤。**
    1. 给出段选择子（放在段选择寄存器中）+偏移量
    2. 若选择了LDT方式，则从GDTR获取GDT首地址，用LDTR中的偏移量做偏移，拿到GDT中的对应LDT描述符
    3. 从对应LDT描述符中获取LDT首地址，用段选择子中的高13位做偏移，拿到LDT中的任务私有段描述符
    4. 如果合法且有权限，用任务私有段描述符中的段首地址加上（1）中的偏移量找到物理地址。寻址结束。



8. **根目录区大小一定么？扇区号是多少？为什么?**
    - 根目录区大小不一定，需要计算。由BPB中的根目录项数（BPB地址17开始的16位）决定根目录区的大小，默认为224，而且文件名过长会在原本的目录项后面立即跟一个LFN项。
    - 在FAT12中，根目录区扇区号是19.（0是引导扇区，1-9是FAT1，10-18是FAT2）
9. **数据区第一个簇号是多少？为什么?**
    - 数据区的第一个簇号是2。
    - 由于FAT表项的序号与簇号一一对应，而FAT表中序号为0和1的表项被设置成了固定值，簇0和簇1就没有存在的意义了，所以数据区起始于簇2.
10. **FAT表的作用?**
    - 用来读取数据区中对应的簇。
    - 文件分配表被划分为紧密排列的若干个表项，每个表项都与数据区中的一个簇相对应，而且表项的序号也是与簇号一一对应的。
    - FAT项的值代表文件的下一个簇号，从而形成一个文件的簇链。
        - 大于等于0xFF8，表示当前簇已经是本文件最后的一个簇
        - 等于0xFF7, 表示它是一个坏簇。
11. **解释静态链接的过程。**
    - **静态链接**是指在编译阶段直接把静态库加入到可执行文件中去，这样可执行文件会比较大。
    - 1.空间和地址分配

      2.符号解析和重定位
12. **解释动态链接的过程。**
    - **动态链接**则是指链接阶段仅仅只加入一些描述信息，而程序执行时再从系统中把相应动态库加载到内存中去。
        1. 动态链接器自举
        2. 装载共享对象
        3. 重定位和初始化
        4. 转交控制权
13. **静态链接相关PPT中为什么使用ld链接而不是gcc**
    - gcc默认使用动态链接的方式连接第三方库。
    - 在ppt中若使用gcc -L ./ -lmylib 来链接，链接程序会去找动态库libmylib.so而不是libmylib.a
14. **linux下可执行文件的虚拟地址空间默认从哪里开始分配。**
    - 32位：0x08048000
    - 64位：0x00400000

2.2 实验相关内容

1. **BPB指定字段的含义**

标识	Offset(hex)	Size	Meaning
BPB_BytsPerSec	0x0B	2	每个扇区字节数
BPB_SecPerClus	0x0D	1	每个簇的扇区数
BPB_RsvdSecCnt	0x0E	2	Boot record占用扇区数
BPB_NumFATs	0x10	1	FAT的数量
BPB_RootEntCnt	0x11	2	根目录最大文件数
BPB_TotSec16	0x13	2	扇区数
BPB_FATSz16	0x16	2	每个FAT占用扇区数

2. **如何进入子目录并输出（说明方法调用）**
    - 调用getChildren()解析该子目录
    1. 根据根目录项获得子目录的第一个簇
    2. 然后遍历簇中的每一个目录项
        1. 使用getDirEntry()解析目录项
            1. 若为文件，getContent()获取内容
            2. 若为子目录，getChildren()解析目录
    3. 获取下一个簇，若当前簇非最后一个簇，回到第2步，否则结束
3. **如何获得指定文件的内容，即如何获得数据区的内容（比如使用指针等）**
    - getContent()方法
    - 遍历簇链中的每一个簇
    - 使用fread提取每一个簇的内容至暂存区contentPerClus
    - 把暂存区的内容累加到该文件节点中的content中
4. **如何进行C代码和汇编之间的参数传递和返回值传递**
    - C代码把参数从右往左压栈，然后把返回地址压栈。
    - 汇编代码通过mov指令把C语言压栈的参数读取出来。
    - 通过栈操作进行参数传递，返回值默认存在eax寄存器中。
5. **汇编代码中对I/O的处理方式，说明指定寄存器所存值的含义**
    - 输入
        - edx 读的长度
        - ecx 存放的位置的地址
        - ebx 指定文件描述符，0 标准输入
        - eax 系统调用号

```
mov     edx, 255        ; number of bytes to read
mov     ecx, sinput     ; reserved space to store our input (known as a buffer)
mov     ebx, 0          ; write to the STDIN file
mov     eax, 3          ; invoke SYS_READ (kernel opcode 3)
int     80h
```

    - 输出
        - edx 写的长度
        - ecx 用于输出的字符串的内存地址
        - ebx 指定文件描述符，1 标准输出
        - eax 系统调用号

```
mov     edx, 13     ; number of bytes to write - one for each letter plus 0Ah (line feed character)
mov     ecx, msg    ; move the memory address of our message string into ecx
mov     ebx, 1      ; write to the STDOUT file
mov     eax, 4      ; invoke SYS_WRITE (kernel opcode 4)
int     80h
```