**问题清单:**

1. 实模式下的寻址⽅式以及实模式的缺陷

实模式下的寻址方式：物理地址=段值\*16+偏移（段在内存中的位置固定）

缺陷：（1）通过改变段寄存器的值，可以不受限制地访问任意一个内存单元，谈不上对系统的保护；

（2）同一个内存地址可由多个段基址加偏移表示，分段地址的比较将复杂化。

2.保护模式下的寻址过程： 段寄存器中存储的是什么? GDT是什么?LDT是什么?LDT和GDT的区别是什么? 如何定位到Descriptor?Descriptor的内容有哪些?

段寄存器中存储的是对应段的Selector选择子，GDT是全局描述符表，LDT是局部描述符表，区别：LDT为各进程独占，每个进程都有属于自己的LDT，而GDT为所有进程共享。

定位到Descriptor的过程：

根据段寄存器中的选择子判断描述符是在GDT还是LDT中，如果是在GDT中，根据GDTR（32位基址及16位段限）及段寄存器中Selector的描述符索引获取Descriptor，如果是在LDT中，先根据GDTR及LDTR中Selector的索引部分获取LDT基址，再根据段寄存器中选择子获取对应描述符。

Descriptor（8个字节）的内容：

段基址（32位）、段界限（20位）、访问权限、描述符类型等

3.选择⼦的作⽤：选择⼦是什么？它的值存放在哪⾥？选择⼦⾥⾯的内容有哪些？为什么偏移地址⼤⼩13位？

Selector选择子，一个两字节的数（16位），最低两位RPL（请求特权等级），第三位TI（表示查表是通过GDT还是LDT进行），最高13位给出了所需描述符在表中的索引，用于描述符的获取。（它的值存放在段寄存器或LDTR）

GDTR中段限为16位，而每个描述符占8个字节，2^16/2^3=2^13,即最多8k个描述符，所以偏移地址大小为13位。

4.描述符的作⽤

描述符作用：

1. 获取段基址
2. 判断访问是否越界
3. 判断访问是否越权

5.GDTR/LDTR的作⽤： GDTR的内容是什么？ LDTR的内容是什么？ 为什么LDT要放在GDT中？

GDTR用于存放GDT在内存中的基址和段限，GDTR是一个48位的寄存器，其中32位表示基址，16位表示段限

LDTR中只存放一个“选择子”，提供对应LDT描述符在GDT中的索引，进而可以获取LDT的内存地址

LDT个数随任务数量变化，不像GDT固定一个，可直接用一个GDTR存储信息，因而将LDT放在GDT中

6.根⽬录区⼤⼩⼀定么?扇区号是多少?为什么？

根目录区大小不是一定的，根目录大小=BPB\_RootEntCnt\*32字节（根目录区一个条目的大小）；

起始扇区号=1+BPB\_FATSz16\*2，FAT12文件系统为19。

7.数据区第⼀个簇号是多少?为什么?

数据区第一个簇号是2，因为FAT文件系统中FAT表的前两项为固定值，导致簇0和簇1没有意义。比如一个应用在1.44M软盘上的FAT12文件系统，FAT前三个字节的值是固定的，分别是0xF0、0xFF、0xFF。

8.FAT表的作⽤?

文件分配表中每个表项都与数据区中的一个簇相对应（表项序号与簇号）， 而FAT项的值代表的是文件的下一个簇号，当值大于或等于0xFF8，说明当前簇已是本文件的最后一个簇，0xFF7则为坏簇。FAT表使文件大小可以超出一个簇，在读取占据超过一个簇（512字节，BPB\_SecPerClus\*BPB\_BytesPerSec）的文件数据时，可通过查询FAT表完成。

9.解释静态链接的过程。

静态链接：静态链接就是在编译链接时直接将需要的执行代码拷贝到调用处，

（使用静态链接生成的可执行文件体积较大，包含相同的公共代码，造成浪费）

10.解释动态链接的过程。

使用这种方式的程序并不在一开始就完成动态链接，而是直到真正调用动态库代码时，载入程序才计算(被调用的那部分)动态代码的逻辑地址。

（这种方式使程序初始化时间较短，但运行期间的性能比不上静态链接的程序）

11.静态链接相关PPT中为什么使⽤ld链接⽽不是gcc。

gcc默认使用动态链接库，这样就达不到静态链接的目的，因而采用ld

12.linux下可执⾏⽂件的虚拟地址空间默认从哪⾥开始分配。

在32位X86架构的Linux系统中，用户进程可执行程序一般从虚拟地址空间0x08048000开始加载。该加载地址由ELF文件头决定，可通过自定义链接器脚本覆盖链接器默认配置，进而修改加载地址。0x08048000以下的地址空间通常由C动态链接库、动态加载器ld.so和内核VDSO(内核提供的虚拟共享库)等占用。