Q:

**有时，继承传统意味着别手蹩脚。x86计算机为了向下兼容，导致启动过程比较复杂。请找出x86计算机启动过程中，被硬件强制，软件必须遵守的两个“多此一举”的步骤（多找几个也无妨），说说它们为什么多此一举，并设计更简洁的替代方案。**

A:

1）x86CPU启动时为了向下兼容16位使用实模式：纯16位无保护执行环境。对于80286或以上的CPU通过A20 GATE来控制A20地址线。 技术发展到了80286，虽然系统的地址总线由原来的20根发展为24根，这样能够访问的内存可以达到2^24=16M,但是Intel在设计80286时提出的目标是向下兼容, 所以在实模式下，系统所表现的行为应该和8086/8088所表现的完全一样，也就是说，在实模式下，80386以及后续系列应该和8086/8088完全兼容仍然使用A20地址线。所以高级芯片为了运行以前的程序，不得不保留实模式。所以说80286芯片存在一个BUG：它开设A20地址线。如果程序员访问100000H-10FFEFH之间的内存，系统将实际访问这块内存。进入实模式多此一举，可以直接进入保护模式。

解决方案：是不向下兼容直接进入32位的保护模式，尽管在Intel 80286手册中已经提出了虚地址保护模式，但实际上它只是一个指引，真正的32位地址出现在Intel 80386上。保护模式本身是80286及以后兼容处理器序列之后产成的一种操作模式，它具有许多特性设计为提高系统的多道任务和系统的稳定性.例如内存的保护，分页机制和硬件虚拟存储的支持。现代多数的x86处理器操作系统都运行在保护模式下。

2）当PC的电源打开后，80x86结构的CPU将自动进入实模式，并从地址0xFFFF0开始自动执行程序代码，这个地址通常是ROM—BIOS中的地址。PC机的BIOS将执行某些系统的检测，并在物理地址0处开始初始化中断向量。此后将启动设备的第一个扇区512字节读入内存绝对地址0x7C00处。因为当时system模块的长度不会超过0x80000字节大小512KB，所以bootsect程序把system模块读入物理地址0x10000开始位置处时并不会覆盖在0x90000处开始的bootsect和setup模块，多此一举的是system模块移到内存中相对靠后的位置，以便加载系统主模块。

解决方案是在保证操作系统启动引导成功的前提下尽量扩大ROM—BIOS的内存寻址范围，以达到不需要读入靠后的位置处。

3）BIOS初始化时，会在物理内存开始处放置大小位1KB的中断向量表，供BIOS的中断使用。这就强制了操作系统的引导程序在加载操作系统的主模块时如果要利用BIOS中断获取一些信息，则主模块的加载位置不能将BIOS的向量表覆盖掉，而操作系统的主模块一般运行时要在内存的开始处（这样主模块中的代码地址也就等于实际的物理地址，便于操作），所以操作系统的引导程序会先将主模块（如Linux0.11中的System模块）读到内存中不与BIOS中断向量表冲突的位置，之后在将主模块移动到内存起始处，将BIOS的中断向量表覆盖掉，这是“多此一举”的。

解决方案是BIOS初始化时将BIOS的中断向量表放到内存中其他实模式下BIOS可以访问到的内存处，这样操作系统引导程序就可以直接将操作系统的主模块读到内存的起始处了。