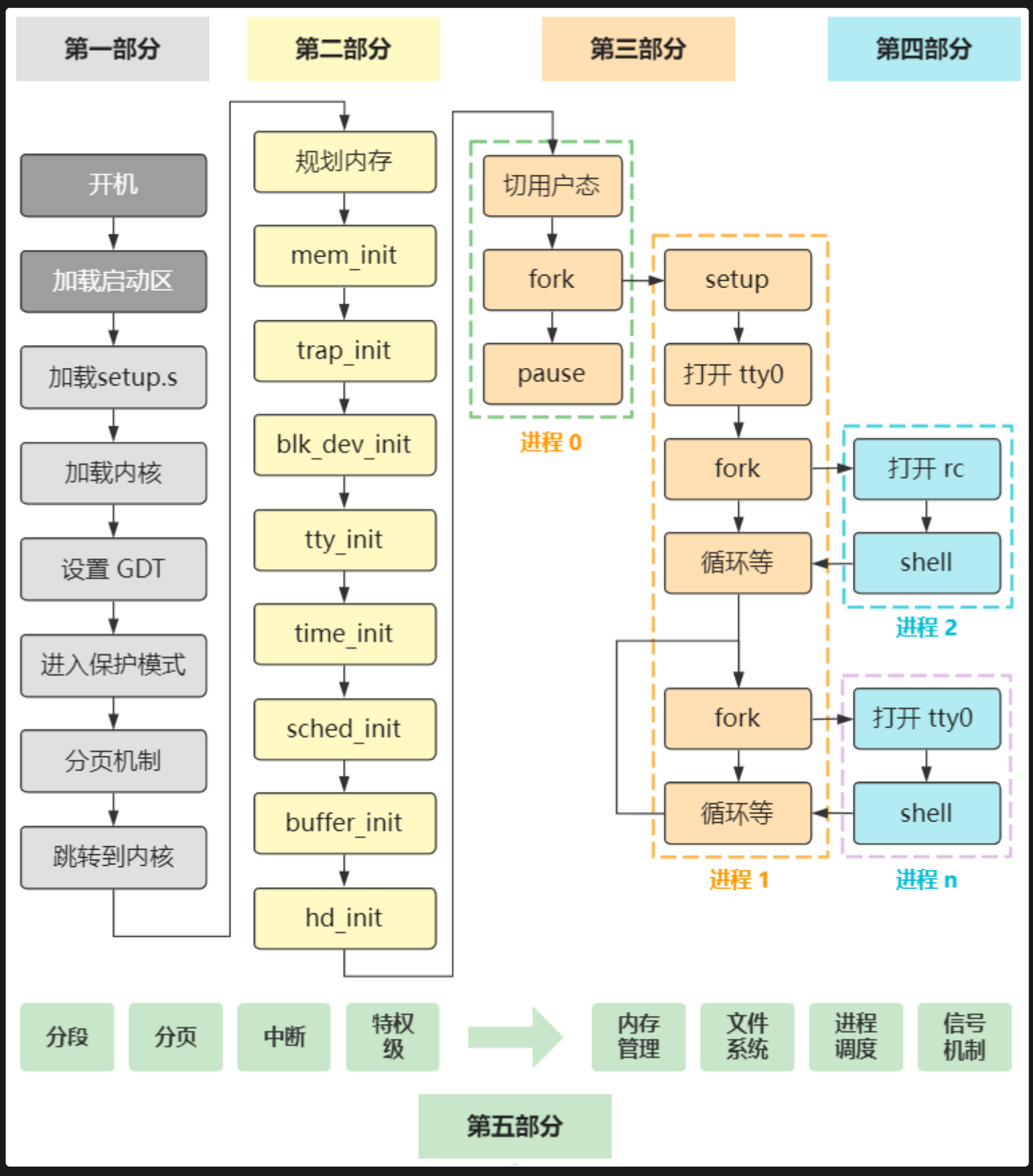
第三回

新读者看这里，老读者直接跳过。

本系列会以一个读小说的心态，从开机启动后的代码执行顺序，带着大家阅读和赏析 Linux 0.11 全部核心代码，了解操作系统的技术细节和设计思想。



你会跟着我一起，看着一个操作系统从啥都没有开始，一步一步最终实现它复杂又精巧的设计，读完这个系列后希望你能发出感叹，原来操作系统源码就是这破玩意。以下是已发布文章的列表，详细了解本系列可以先从开篇词看起。

开篇词

第一回 最开始的两行代码

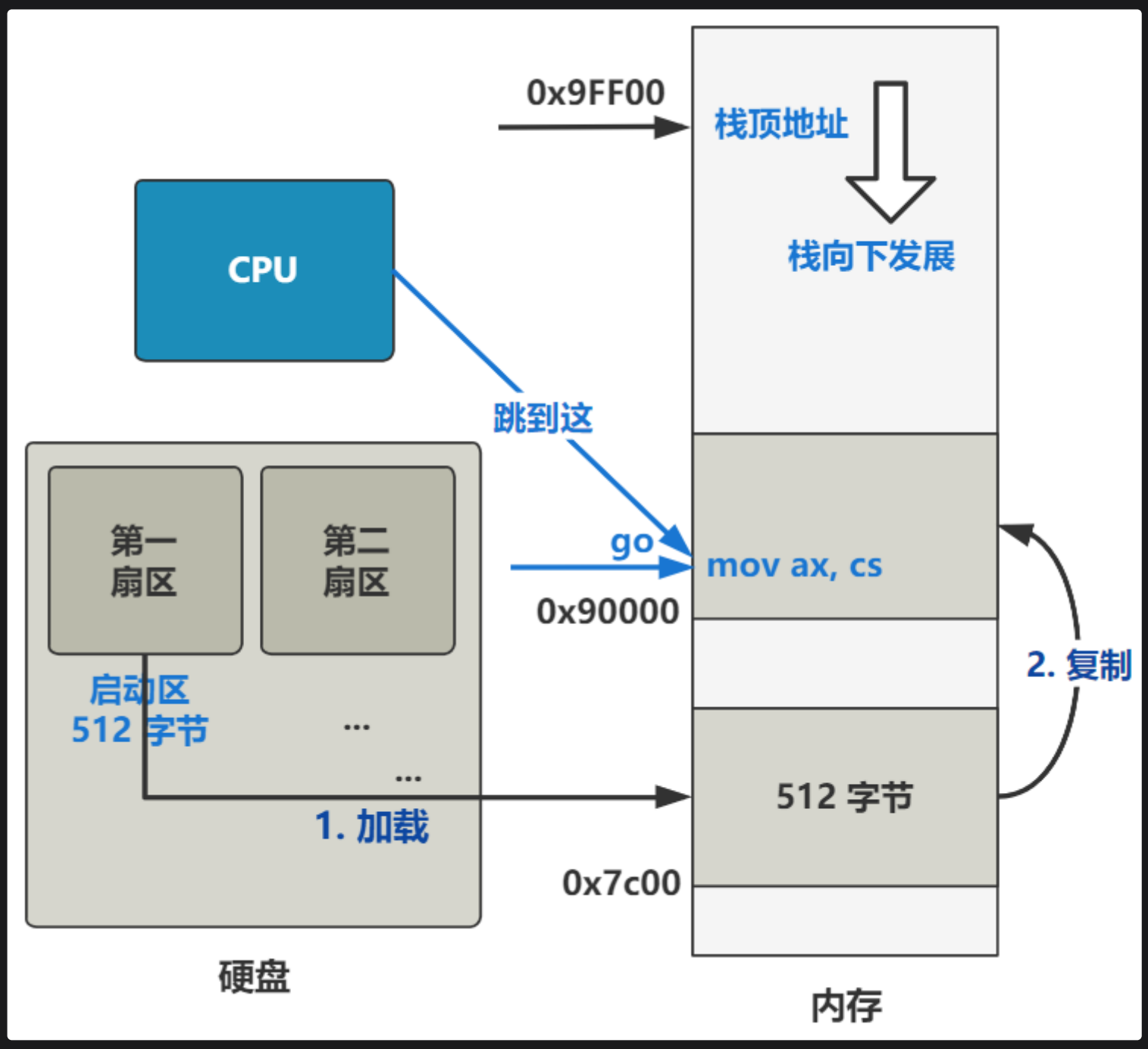
第二回 自己给自己挪个地儿

本系列的 GitHub 地址如下

https://github.com/sunym1993/flash-linux0.11-talk

------- **正文开始** -------

书接上回，上回书咱们说到，操作系统的代码最开头的 512 字节的数据，从硬盘的启动区先是被移动到了内存 **0x7c00** 处，然后又立刻被移动到 **0x90000** 处，并且跳转到此处往后再稍稍偏移 **go** 这个标签所代表的偏移地址处。



那我们接下来，就继续把我们的目光放在 go 这个标签的位置，跟着 CPU 的步伐往后看。

```
    go: mov ax,cs    mov ds,ax    mov es,ax    mov ss,ax    mov sp,#0xFF00
```

全都是 mov 操作，那好办了。

这段代码的直接意思很容易理解，就是把 **cs** 寄存器的值分别复制给 **ds**、**es** 和 **ss** 寄存器，然后又把 0xFF00 给了 **sp** 寄存器。

回顾下 CPU 寄存器图。

CPU 中的关键寄存器

cs 寄存器表示**代码段寄存器**，CPU 当前正在执行的代码在内存中的位置，就是由 cs:ip 这组寄存器配合指向的，其中 cs 是基址，ip 是偏移地址。

由于之前执行过一个段间跳转指令，还记得不？

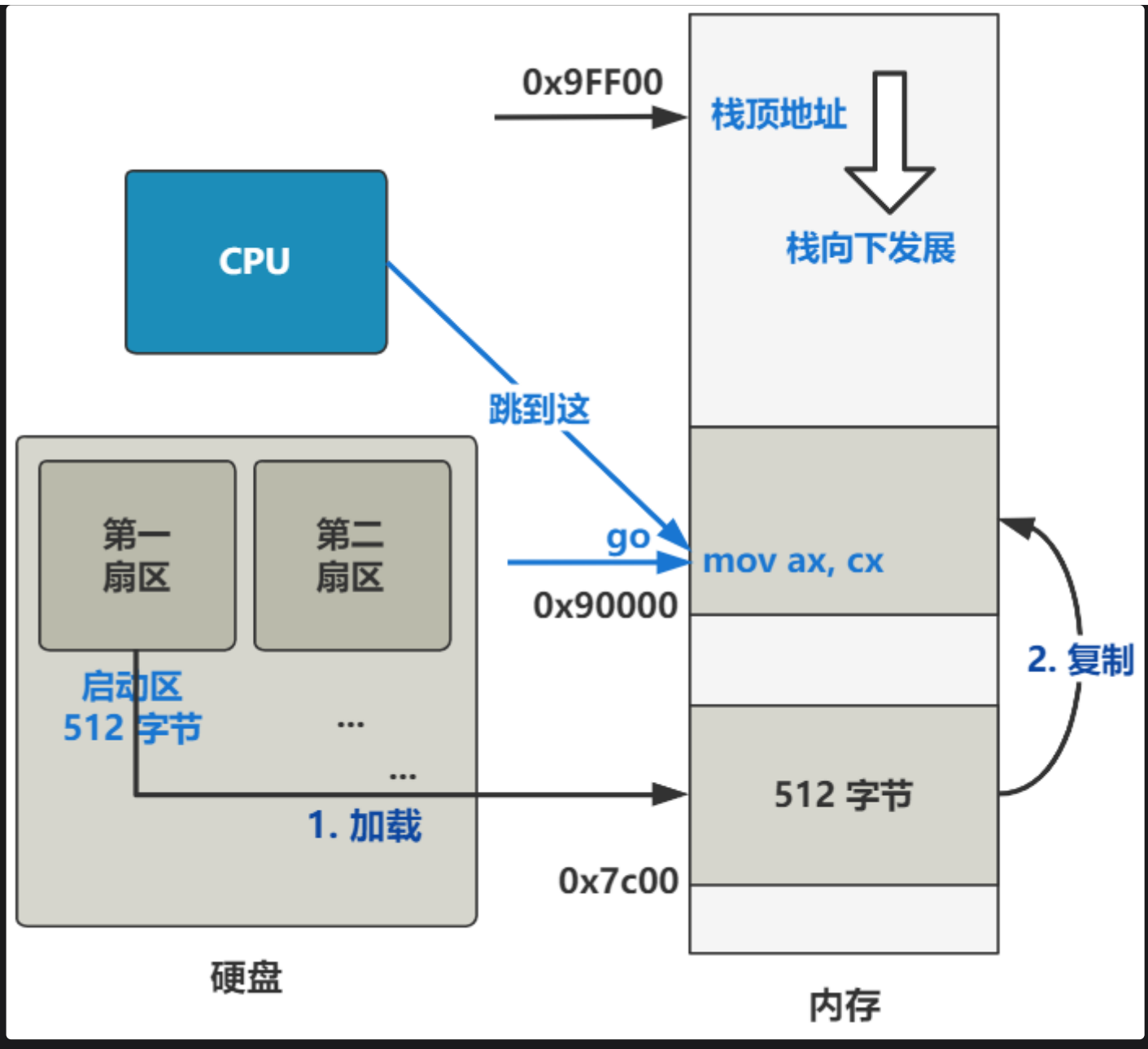
```
jmpi go,0x9000
```

所以现在 cs 寄存器里的值就是 **0x9000**，ip 寄存器里的值是 **go** 这个标签的偏移地址。那这三个 mov 指令就分别给 ds、es 和 ss 寄存器赋值为了 0x9000。

ds 为数据段寄存器，之前我们说过了，当时它被复制为 **0x07c0**，是因为之前的代码在 0x7c00 处，现在代码已经被挪到了 0x90000 处，所以现在自然又改赋值为 **0x9000** 了。

es 是扩展段寄存器，仅仅是个扩展，不是主角，先不用理它。

ss 为**栈段寄存器**，后面要配合栈基址寄存器 sp 来表示此时的栈顶地址。而此时 sp 寄存器被赋值为了 **0xFF00** 了，所以目前的栈顶地址就是 **ss:sp** 所指向的地址 **0x9FF00** 处。



其实到这里，操作系统的一些最最最最基础的准备工作，就做好了。都做了些啥事呢？

第一，代码从硬盘移到内存，又从内存挪了个地方，放在了 **0x90000** 处。

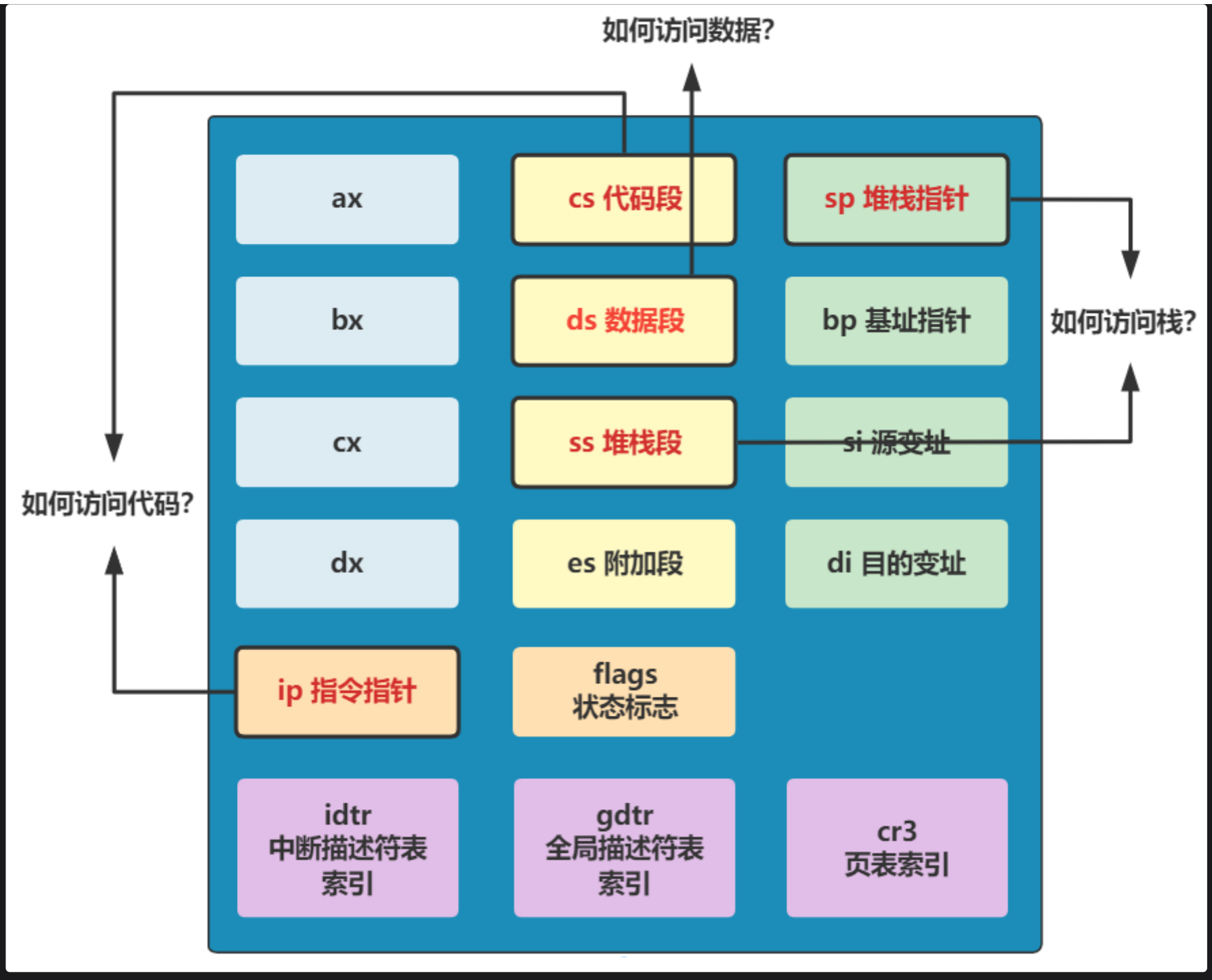
第二，**数据段寄存器 ds** 和**代码段寄存器 cs** 此时都被设置为了 0x9000，也就为跳转代码和访问内存数据，奠定了同一个内存的基址地址，方便了跳转和内存访问，因为仅仅需要指定偏移地址即可了。

第三，栈顶地址被设置为了 0x9FF00，具体表现为**栈段寄存器 ss** 为 0x9000，**栈基址寄存器 sp** 为 0xFF00。栈是向下发展的，这个栈顶地址 0x9FF00 要远远大于此时代码所在的位置 0x90000，所以栈向下发展就很难撞见代码所在的位置，也就比较安全。这也是为什么给栈顶地址设置为这个值的原因，其实只需要离代码的位置远远的即可。

做好这些基础工作后，接下来就又该折腾了其他事了。

总结拔高一下，这一部分其实就是把**代码段寄存器 cs**，**数据段寄存器 ds**，**栈段寄存器 ss** 和**栈基址寄存器 sp** 分别设置好了值，方便后续使用。

再拔高一下，其实操作系统在做的事情，就是给如何访问代码，如何访问数据，如何访问栈进行了一下**内存的初步规划**。其中访问代码和访问数据的规划方式就是设置了一个**基址**而已，访问栈就是把**栈顶指针**指向了一个远离代码位置的地方而已。



所以，千万别多想，就这么点事儿。那再给大家留个作业，把当前的内存布局画出来，告诉我现在 **cs、ip、ds、ss、sp** 这些寄存器的值，在内存布局中的位置。

好了，接下来我们应该干什么呢？我们回忆下，我们目前仅仅把硬盘中 512 字节加载到内存中了，但操作系统还有很多代码仍然在硬盘里，不能抛下他们不管呀。

所以你猜下一步要干嘛了？

后面的世界越来越精彩，欲知后事如何，且听下回分解。