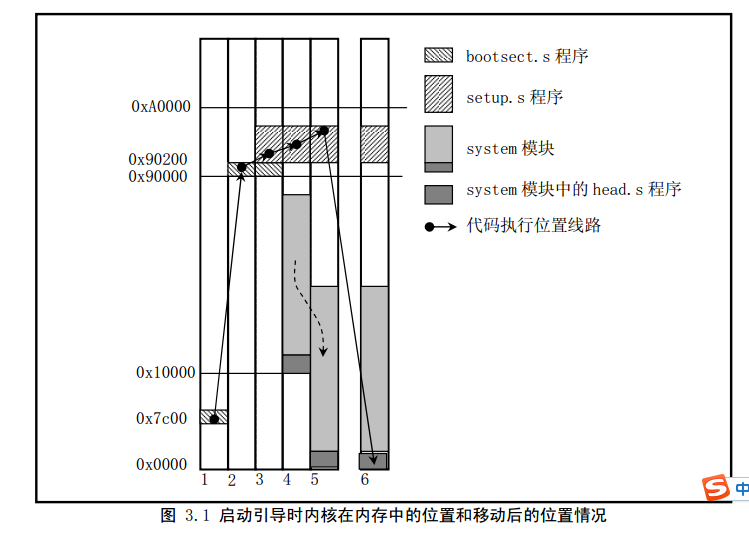
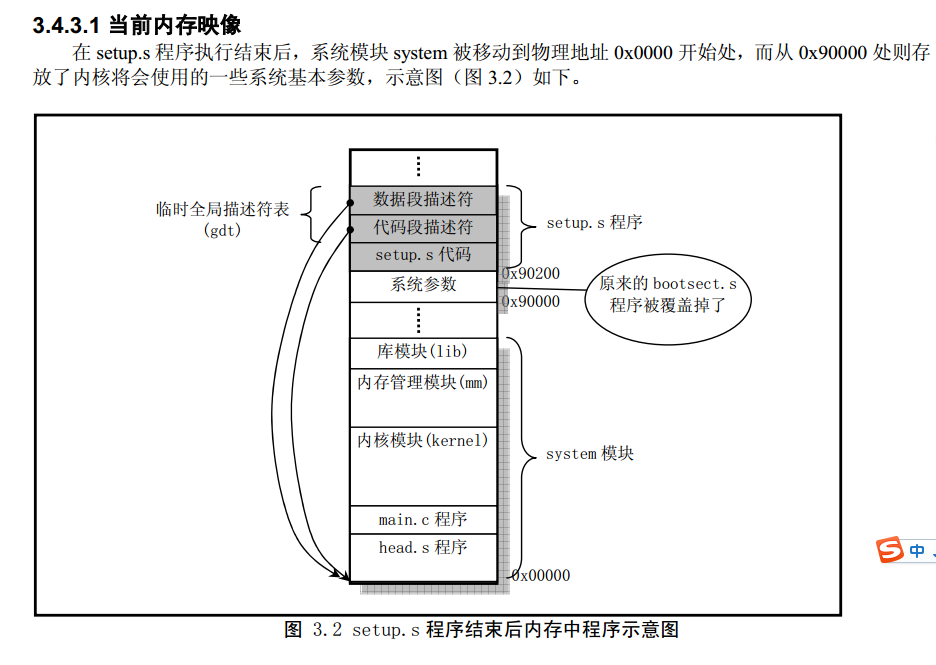
Linux 的最最前面部分是用 8086 汇编语言编写的(boot/bootsect.s)，它将由 **BIOS 读入到内存绝对地址0x7C00(31KB)**处，当它被执行时就会把自己移到绝对地址 **0x90000(576KB)**处，并把启动设备中后 **2kB 字节代码(boot/setup.s)**读入到内存 **0x90200** 处，而内核的其它部分（system 模块）则被读入到从地址 **0x10000**开始处， 因为当时 **system 模块的长度不会超过 0x80000 字节大小（即 512KB），** 所以它不会覆盖在 0x90000处开始的 bootsect 和 setup 模块。随后将 system 模块移动到内存起始处，这样 **system 模块中代码的地址也**

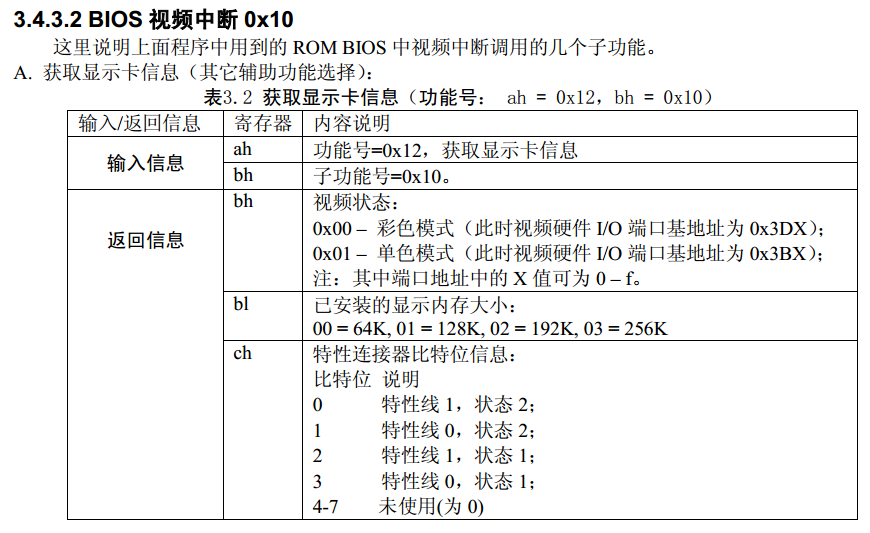
**即等于实际的物理地址**。便于对内核代码和数据的操作。图 3.1 清晰地显示出 Linux 系统启动时这几个程序或模块在内存中的动态位置。其中，每一竖条框代表某一时刻内存中各程序的映像位置图。在系统加载期间将显示信息"Loading..."。然后控制权将传递给 **boot/setup.s** 中的代码， 这是另一个实模式汇编语言程序。

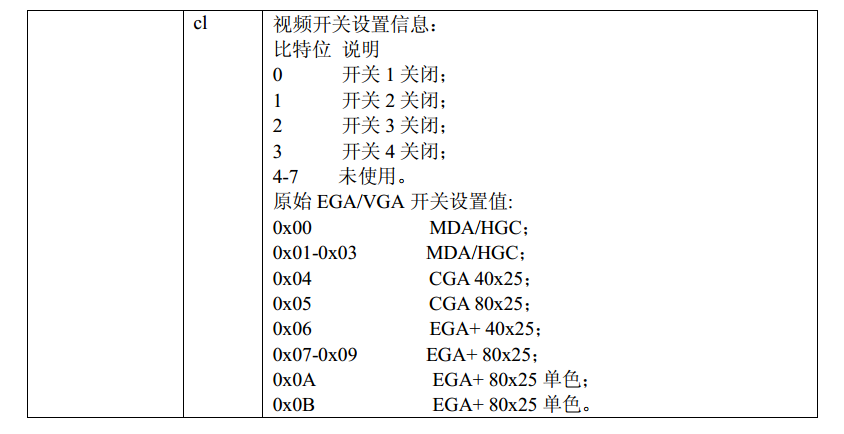
启动部分识别主机的某些特性以及 vga 卡的类型。如果需要，它会要求用户为控制台选择显示模式。

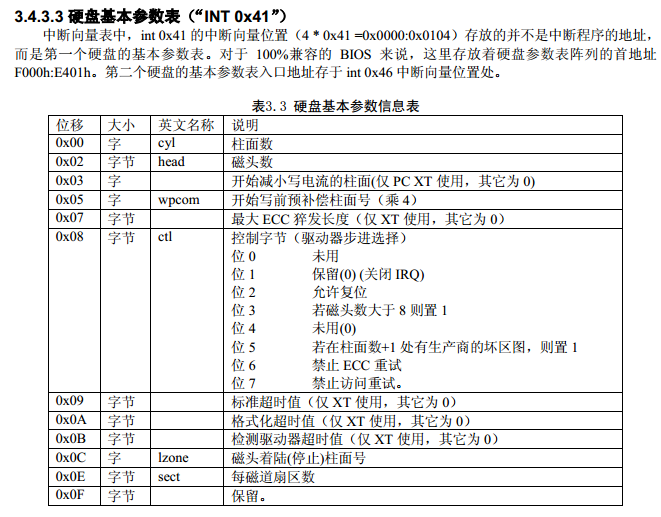
然后将整个系统从地**址 0x10000 移至 0x0000** 处，**进入保护模式并跳转至系统的余下部分（在 0x0000 处）。**此时所有 32 位运行方式的设置启动被完成: **IDT、 GDT 以及 LDT 被加载，处理器和协处理器**也已确认，

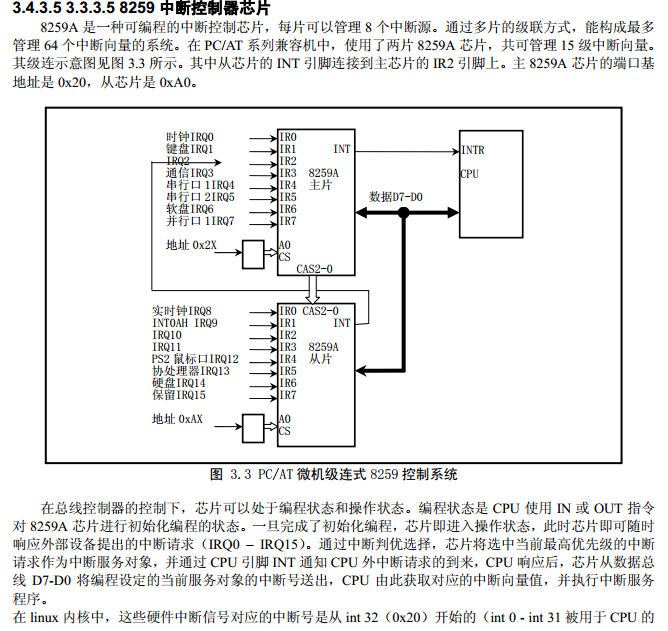
分页工作也设置好了；最终调用 **init/main.c 中的 main()程序**。上述操作的源代码是在 **boot/head.S** 中的，这可能是整个内核中最有诀窍的代码了。注意如果在前述任何一步中出了错，计算机就会死锁。在操作系统还没有完全运转之前是处理不了出错的。

 **bootsect.s 代码是磁盘引导块程序**，驻留在磁盘的第一个扇区中（引导扇区， 0 磁道（柱面）， 0 磁头，第 1 个扇区）。在 PC 机加电 ROM BIOS 自检后，引导扇区由 BIOS 加载到内存 **0x7C00** 处，然后将自己移动到内存 **0x90000** 处。该程序的主要作用是首先**将 setup 模块（由 setup.s 编译成）从磁盘加载到内存，紧接着 bootsect 的后面位置（0x90200） ,然后利用 BIOS 中断 0x13 取磁盘参数表中当前启动引导盘的参数**，接着在屏幕上显示“Loading system...”字符串。再者将 **system 模块从磁盘上加载到内存 0x10000** 开始的地方。随后确定根文件系统的设备号，若没有指定，则根据所保存的引导盘的每磁道扇区数判别出盘的类型和种类（是 1.44M A 盘？）并保存其设备号**于 root\_dev(引导块的 0x508 地址处)**，最后长跳转到 **setup 程序的开始处（0x90200）**执行 setup 程序。









陷阱中断），也即中断号范围是 int32 -- int 47。

**3.4.3.6 Intel CPU 32 位保护运行模式**

Intel CPU 一般可以在两种模式下运行，即实地址模式和保护模式。早期的 Intel CPU（8088/8086）只能工作在实模式下，某一时刻只能运行单个任务。对于 Intel 80386 以上的芯片则还可以运行在 32 位保护模式下。在保护模式下运行可以支持多任务；支持 4G 的物理内存；支持虚拟内存；支持内存的页式管理和段式管理；支持特权级。

虽然对保护模式下的运行机制是理解 Linux 内核的重要基础，但由于篇幅所限，对其工作原理的简单介绍可以参考书后的附录。但仍然建议初学者能够使用书后列出相关书籍，作一番仔细研究。为了真正理解 setup.s 程序和下面 head.s 程序的作用，起码要先明白段选择符、段描述符和 80x86 的页表寻址机制。

3.4.3.7 内存管理寄存器  
Intel 80386 CPU 有 4 个寄存器用来定位控制分段内存管理的数据结构：

GDTR （Global Descriptor Table Register）全局描述符表寄存器；

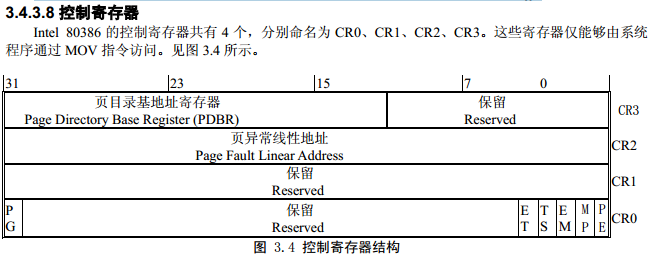
LDTR （Local Descriptor Table Register）局部描述符表寄存器；

这两个寄存器用于指向段描述符表 GDT 和 LDT。这两个表是用于内存的分页管理，参见附录中的描述。

IDTR （Interrupt Descriptor Table Register）中断描述符表寄存器；

这个寄存器指向中断处理向量（句柄）表（IDT）的入口点。所有中断处理过程的入口地址信息均存放在该表中的描述符表项中。

TR （Task Register）任务寄存器；

该寄存器指向处理器定义当前任务（进程）所需的信息，也即任务数据结构 task{}。

控制寄存器 CR0 含有系统整体的控制标志，它控制或指示出整个系统的运行状态或条件。其中：

PE – 保护模式开启位（Protection Enable，比特位 0）。如果设置了该比特位，就会使处理器开始在保护模式下运行。

MP – 协处理器存在标志（Math Present，比特位 1）。用于控制 WAIT 指令的功能，以配合协处理的运行。

EM – 仿真控制（Emulation，比特位 2）。指示是否需要仿真协处理器的功能。

TS – 任务切换（Task Switch，比特位 3）。每当任务切换时处理器就会设置该比特位，并且在解释协处理器指令之前测试该位。

ET – 扩展类型（Extention Type，比特位 4）。该位指出了系统中所含有的协处理器类型（是 80287 还是80387）。

PG – 分页操作（Paging，比特位 31）。该位指示出是否使用页表将线性地址变换成物理地址。参见第 10章对分页内存管理的描述。

CR2 用于 PG 置位时处理页异常操作。 CPU 会将引起错误的线性地址保存在该寄存器中。CR3 同样也是在 PG 标志置位时起作用。该寄存器为 CPU 指定当前运行的任务所使用的页表目录