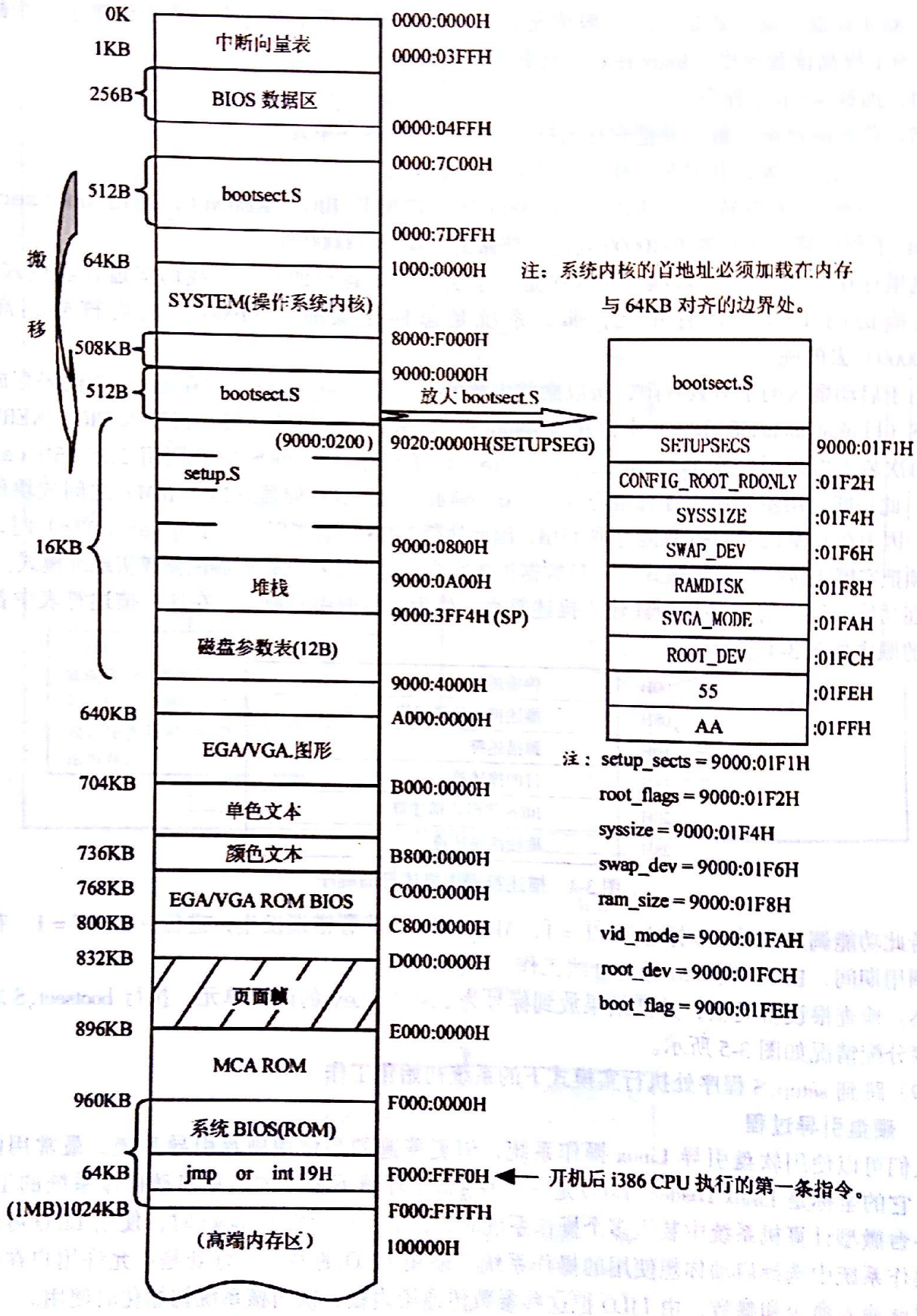

图 3-5　执行 bootsect.S 主引导程序后的内存分配图

第4章　启动 Linux 内核

当系统调用 start_Kernel（）函数时，标志着核心程序开始启动并在保护模式下执行 Linux 内核的初始化过程。首先，核心将对内存进行检测，以便知道系统中一共装有多少内存，内核占用了多少，还剩多少可作为动态内存供核心与普通进程使用[5]。

除此之外，start_Kernel（）调用各方面的函数初始化内核的一些重要数据结构。另外，核心还要对系统中所安装的各种硬件包括外围设备进行检测。核心将通过与设备所连接的总线及其相应的设备驱动程序来获得设备信息，并对检测结果作出标记写入相应单元，供系统使用。一般情况下，核心将报告它应该找到但却由于种种原因没有找到的设备。

在核心自身的初始化完成之后，start_Kernel（）最后将调用 kernel_thread（）函数，创建系统 1 号进程，1 号进程又叫做 init 进程。它用来启动 shell 命令文本的执行。操作系统为了向用户提供各种服务，通常还要生成一些提供这些服务的内核线程。内核线程定期被激活，为系统中的用户提供一些特定的服务。

这些内核线程是由 init 函数生成的。在各种内核线程开始工作之后，init 函数通过执行 exec（）函数变成 init 进程，剩下的事情全部交给 init 进程去做。到此为止操作系统的启动过程就告一段落，此时终端屏幕上会出现系统版本号，显示登录提示符或出现一个图形窗口，以便让用户依照各自的目标，正常使用计算机系统。

4.1　初始化系统内核

当内核代码被装入内存并解压缩之后，系统已在低端对一些重要的硬件设备，如内存管理单元（MMU），进行了初始化设置[4]。此时程序控制转移到 start_kernel（init/main.c）处，继续初始化系统内核的各种重要数据结构，它几乎做了初始化内核的所有工作。start_kernel 函数的执行流程如图 4-1 所示。

初始化内核系统所使用的各种函数标记有__init 或__initfunc 或__initdata 符号。gcc 编译器编译源代码时将带这些标记的函数放在内存的特殊位置上，在内核系统初始化结束后，这些函数所占用的内存就被释放，可作为动态内存供其他程序使用。

4.1.1　初始化与体系结构相关的硬件数据结构

对硬件数据结构的初始化过程如下：

（1）首先调用 printk（）函数在屏幕上显示 Linux 内核的版本号以及编译内核所使用的 gcc 版本号、启用时间等。如果系统在机器上启动失败，将为用户提供一些参考信息。

（2）调用 arch/i386/kernel/setup.c 中的 setup_arch（）函数，初始化系统主板上的各个集成电路控制器。最后在 command line、memory_start 和 memory_end 中返回结果。因为在紧接着的初始化过程中，同样有一些程序代码和数据要占用内存，因此在各初始化函数执行结束后，调整相应内存的边界值，并返回该值供以后的初始化函数使用。setup_arch（）函数完成的工作如图 4-1 所示：