## main函数之前

1. BIOS（这部分固化在ROM中，没有代码）

作用：设置中断向量表，中断服务程序，BIOS数据段，读入bootsect代码。

过程：

加电即进入16位实模式，将段寄存器cs强制置为0xF000，指令指针寄存器ip置为0xFFF0，则指令从0xFFFF0出开始执行，即BIOS代码的位置。在内存0x00000处开始使用1K（0-0x3FFF）空间存储中断向量表，总共256项，每一项4字节，前两字节表示段地址，后两字节表示段内偏移。后面紧接256字节（0x400-0x4FF）的BIOS数据区。从0x0E05B开始的中断服务程序。

使用int 0x19中断从硬盘加载一个扇区256B（bootsect.s）到内存的0x7C00处。到此BIOS将运行权限交由bootsect执行。

1. bootsect.s

作用：规划内存、加载setup.s、加载system

过程：将自身代码从0x7c00处移动到0x90000处；中断0x13将4个扇区的setup.s程序读入到bootsect之后；再将240扇区的system模块读入到0x10000位置（setup.s和system之间隔着一个栈区）。bootsect代码执行完成之后，接着执行setup.s

规划内存：

SETUPLEN **=** 4 **;**// setup程序的扇区数（setup－sectors）值

BOOTSEG **=** 07c0h **;**// bootsect的原始地址（是段地址，以下同）

INITSEG **=** 9000h **;**// 将bootsect移到这里

SETUPSEG **=** 9020h **;**// setup程序从这里开始

SYSSEG **=** 1000h **;**// system模块加载到10000(64kB)处.

ENDSEG **=** SYSSEG **+** SYSSIZE **;**// 停止加载的段地址

移动bootsetc到位置0x90000处，int 0x13加载setup到0x90020处，int 0x13 加载system到0x10000处。

1. setup.s

作用：读取系统运行所需机器参数并覆盖bootsect区域、关闭中断->移动system到0x00000->设置IDTR和GDTR->打开32位模式->打开保护模式->跳转到head.s

1. head.s

作用：重新设置IDT和GDT、main地址压栈、设置页目录表和页表、设置CR3和CR0 PG位

## main函数进程0

1. 设置根设备和硬盘
2. 规划缓冲区、虚拟盘、主内存

虚拟盘—>blk\_dev[7]

主内存—>mem\_map

1. 中断服务例程挂接

IDT:set\_trap\_gate:15 0

1. 设置请求项结构

requests[32]

1. tty\_init等人机交互设置

set\_intr\_gate:14 0

1. time\_init开机时间
2. 初始化进程0

task[64]

init\_task:tss,ldt-->GDT

开启多进程

系统调用：set\_system\_call:15 3

1. 初始化缓冲区管理

buffer\_init()🡪

buffer\_head

buffer\_map[307]-🡪内核数据区

1. 挂接软硬盘

hd\_init,floopy\_init

1. 开中断

sti()

1. 0特权到3特权的翻转

move\_to\_user\_mode->

使用模拟终端，人为将ss esp eflags cs eip按顺序压栈，iret；实现特权0到特权3的翻转。