从最底层开始 用实模式 逻辑地址 段:偏移

标号可以不打冒号

masm编译后文件在f5c+10 f6c:0处开始执行用户代码，之前的是文件头 但是还包括系统代码，所以起点要好好找

可以编译成exe后删掉文件头 从55AA往上取，引导可能从硬盘或者软盘，数据都是一样的

关于系统启动 硬盘主引导扇区 0面 0道1 加载到内存0x7c00 ,不同厂家硬盘格式不同

主引导区 512字节 最后两字节如果是0x55和0xAA ，引导扇区才有效 系统检测到这两个 就会自动跳转到0x7c00开始执行

必须使标志处于最后两个字节 伪指令 times 重复执行（不是循环是填写）

vdi v 固定尺寸 访问硬盘:chs模式 磁头 磁道 扇区 读/写 逻辑块模式lba vhd

虚拟硬盘读写的工具

虚拟机尝试底层编程

汇编编译后再存入源程序啊

显卡 显存 从显存读取 10来显示画面 真彩色 24个比特 三个字节 对应一个像素

需要算字符像素点位置

直接将ascii码放到显存 文本模式 ，图像模式

显存直接映射到内存空间中 F0000到FFFFF是bios提供 A000到EFFFF是外围设备

文本模式是80\*25 B8000~BFFF是留给显卡显示文本的

Mov ascii码 一些代码是用来控制的，比如响铃 换行 回车 文件头文件尾 确认

每个字符两个字节 一个ascii码 一个是颜色 前景背景 KRGBIRGB K闪烁 I亮度

显示数字， 需要把数字转为自符串 dd dw db

段，默认是ds 当前段 我们设置自己的地址可以用es

引导模式下，Cpu会不断取硬盘指令，所以要避免取到数据应该设置无限循环

Lba连续直写模式

虚拟机的提示都关掉，不显示 调试 断点 bochs可以用来调试，在没有系统的情况下

八、大部分都是无用知识 本章研究如何加载引导扇区 关于段

把主引导扇区改造成应用程序加载器，从硬盘找到执行文件加载，模拟操作系统

外围设备

1、段的定义 段名称 入口段地址dd声明

加载器loader和用户程序 先要检查内存哪里空，然后从硬盘找到文件 16字节对齐

要知道文件长度，入口点 段重定位表

入口点cs:ip，在加载到内存后需要重新计算 段重定位，重新计算各段位置

引导区代码 访问外部设备 输入 输出 模拟信号 usb 总线，控制芯片，输入输出控制设备集中器（ich/南桥），pci总线

端口，i/o接口 每个Io设备都可能有几个端口

端口号 硬盘读写靠扇区 chs模式 lba 逻辑扇区 lba28 lba48 读取扇区数1f2，起始扇区号 分成四段写入1f3 1f4 1f5 1f6 请求读 20 1f7 1f0数据接口，就绪的时候连续读写 1f1错误，最后一次

过程调用：子程序，子例程，例程 都是一样的 call ret 写个硬盘读写程序 近调用远调用 跨段 段地址:偏移地址、

2、加载用户程序 获取大小，读硬盘到内存 读写以扇区为单位

3、重定位用户程序 算每段的段地址 左移 右移

4、跳转执行用户程序 jmp

5、用户程序 先初始化段寄存 访问自己的段 修改栈ds es等

显示字符串 回车 换行 ascii

屏幕光标控制 写入显存 vga文本模式 光标寄存器（显卡）

取光标 通过显卡的索引寄存器 3d4 14 15光位置 读写数据端口 3d5

换行和回车 显卡不认识，必须自己写程序移动光标

显示字符 滚动屏幕 重置光标

段间转移ret retf 访问数据段

返回加载器

九、中断和动态时钟

外部中断NMI,INTR 两根信号线 非屏蔽 可屏蔽 if位 cli sti

中断嵌套，级别更高的中断可以停下当前的中断的处理程序

2、时钟 cmos ram rtc时钟电路 定时发送中断 处理程序在屏幕显示时间

定时中断处理程序和主程序独立，分别执行 hlt指令（使处理器停机，中断会唤醒，比无限循环效率高）

中断向量表ivt

X86新指令

内部中断，软中断 int3 cc 调试 bios中断 调用bios中的程序 int 16键盘

Ffff以下是加载器和堆栈 000以上是bios和外围设备 1000~9ffff

主引导扇区程序

十、

局部性原理：时间：执行过的指令不久之后可能再执行，访问过的数据可能再访问，空间：访问过的内存单元，附近也可能访问

分页，页框替换，虚拟页，物理页

页表 段表

16位分段

全局描述符表 GDT

32位保护程序 IA-32 32位保护模式 内存也是分段 但是可以视为一个段，平坦模式 0~4G

16位模式 32位模式 eflags

16位模式程序可以访问任何地址 32位程序加载时先定义段，然后允许段 ，段界限，特权级别，类型

Es ss cs ds此时保存的不是段，而是段选择子 段寄存器中包含64位的描述符高速缓冲器

真正的段地址 在缓存器中

32位段长无限制 虚拟86模式，可以在保护模式下执行多个8086程序

每次可读写 4 8个字节 双字或者4字 可以用32位地址线访问4G内存

多任务：内存分配和回收 分段模式下程序长度不定，分配碎片化 ia-32支持分页，固定4k的页

线性地址+页 =物理地址 每个任务都有4G的虚拟内存空间

2、现代cpu 流水线，指令操作拆分，给独立单元分别执行 多级流水线，分别分开，每个时钟周期完成几项任务，实现每周期执行一条指令（不能提前执行下一条，除非是不同线程）

高速缓存 寄存器用了触发器，纳秒级 静态存储器sram成本太高 内存是用电容和晶体管做的，动态存储器dram 几十纳秒 硬盘毫秒级

为了不让cpu等待内存和硬盘 高速缓存是sram,源于局部性原理，先调入一块数据到高速缓存， 检索内存数据，缓存已有就在缓存读hit 没有就重新用高速缓存取一块内存miss

高速缓存以块为单位 不用cpu直接读内存，这样反而有可能损失一定时间等待

乱序微操作

寄存器重命名， 把相同寄存器 换到其他临时寄存器，然后重新还给原来的寄存器（隐退）

分支目标预测，存在跳转，可能流水线后面的就无效了 分支目标缓存器btb中有预测结果，下一次执行时，预测失败回退

流水线 微操作 寄存器重命名

32位寻址 默认都是32位 立即数 寄存器 []段内偏移

指令前缀

十一、保护模式

实模式可以修改任何内存单元，实现应用隔离需要保护模式

全局描述符表gdt

全局描述符表寄存器GDTR

段寄存器 ：选择器+描述符缓存器 控制寄存器cr0 段选择子 lgdt

保护模式:段+偏移 需要登记 段描述符 8字节 集中存放的描述符组成描述表

寄存器32位 保存gdt地址 16位保存表大小 实模式只能访问1M内存，保护模式之后可以更换gdt位置

段描述符 gdt中为要访问的段创建描述符 段基地址 段界限 标志 g s dpl p d/b l type avl

1、安装段描述符，加载gdtr lgdt指令 [] 线性地址转段地址

A20地址线问题 替代的a20门控制 0x92端口 A20屏蔽

2、保护模式内存访问 模式切换开关 cr0寄存器 cr1~cr8 cr0第一位1进入保护模式 pe位

保护模式后 bios中断不能用，需要重新设置中断环境，关中断

实模式段左移四位+偏移=20位地址 32位有 fs gs段寄存器 前16位和实模式一样，隐藏的描述符高速缓存器 段选择器中传送的是描述符表的索引号+TI描述符表指示器+RPL 特权 传送到缓存器成为线性32位地址

3、切换的 清理工作 刷新段寄存器 流水线清空 （保护模式下对代码解释不同）

保护模式下cs 不能用mov改 用jmp指令

4、保护模式栈

5、加电时的段寄存器 6、设置pe后的段寄存器

十二、存储器保护

保护模式下，代码段不可写 需要修改就要安装新描述符，定义为可读可写的数据段，称为别名

两个程序共享内存，可以创建描述符指向同一段

3、修改段寄存器 jmp 选择子ti位

4、地址变换的保护 检验地址有效性

5、栈操作的保护

6、数据访问的保护，向上扩展

7、别名访问 字符排序

十三、程序动态加载执行

操作系统加载程序 主引导->内核->用户程序 提供例程给用户调用（API）

内核 初始化代码 内核代码段 数据段 公共例程段

初始化 从bios接管处理器和硬件 安装段描述 从硬盘加载内核其他部分形成各段

数据段开辟空间 例程给双方调用 内核头包含各段位置，方便初始化

1、主引导加载内核 声明 内核硬盘位置，加载的物理内存地址 安装描述符配置几个段

2、给内核各段安装描述符 lgdt进入

3、内核中执行 cpuid 指令

4、加载用户程序

十四、任务和特权保护

任务（task或者进程?） ldt

十四、任务和特权级保护

十五、任务切换

十六、分页和动态页面分配

十七、中断和异常的处理 ，抢占式多任务 中断和异常中断

中断描述符表 idt 中断门 陷阱门 平坦内存模型 硬件中断实现抢占

中断处理/异常处理； 异常就是内部中断：程序错误异常，机器检查异常，软件引发异常 int 0 int3 分为故障，陷阱trap，终止abort 陷阱用于调试

保护模式下的中断 0 1 2 3

实模式 中断向量表 vit 保护模式用中断描述符表:idt 中断描述符表寄存器 idtr