**《30天自制操作系统》读书笔记(3) 引入C语言**

这一次的学习相当曲折, 主要是因为粗心, Makefile里面的错误导致了文件生成出现各种奇奇怪怪的问题, 弄得心力交瘁, 因此制作过程还是尽量按着作者的路子来吧.

作者提供的源码的注释在中文系统下是乱码, 而且代码的分隔用了两个Tab, 在这里要处理一下:

:%s/;.\*//g 删除所有的注释;

:%s/\t\t/\t 把两个Tab替换为一个Tab;

要让作者的nas文件和asm文件拥有相同的语法规则, 在\_vimrc文件的最后一行添加

au BufNewFile,BufRead \*.nas set filetype=asm

* **真正的IPL**

之前我们写的只是软盘启动区代码, 所有代码只能放在512字节里, 根本不够用,

所以我们要利用这一段代码读入软盘的其他内容, 并将控制权移交给它, 所以叫做IPL(Initial Program Loader).

* **需要的知识点:**

**int 0x13 磁盘中断**

AH = 0x02 ; 读盘

AH = 0x03 ; 写盘

AH = 0x04 ; 校验

AH = 0x0c ; 寻道

AH = 0x00 ;大概是重置磁盘, 配合DH = 0x00 使用

AL = 连续处理的扇区数

CH = 柱面号

CL = 扇区号

DH = 磁头号

DL = 驱动器号

ES:BX = 缓冲地址, 寻道及校验不使用

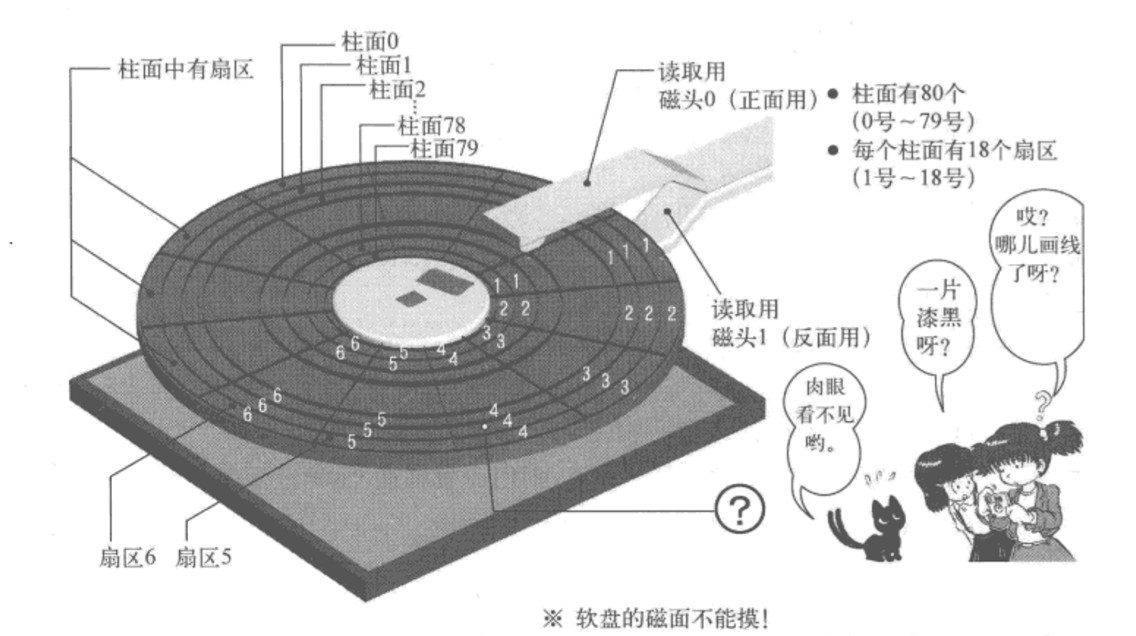
C = 0 没有错误, AH = 0;

C = 1 错误号码存入AH.

这里只会用到0x02和0x00.

**软盘的结构:**

用柱面(C)-磁头(H)-扇区(S) 来标识软盘上的扇区, 我们的启动扇区的标号是C0-H0-S1, 值得注意的是下一个扇区是C0-H0-S2, 这个例子里作者要读入10个柱面, 从C0-H0-S1 读到 C9-H1-S18, 这样算来应该是18\*2\*10 = 360 个扇区, 大小是360\*512/1024 = 180 KB, 从我们现在小的可怜的需求看来, 180KB绝对够了.(按这样算的话不是一个柱面有36个扇区吗? 此处不解).



以下的代码能将软盘除启动区外开始的180KB的空间读入内存中, 起始地址是0x8200, 占用的空间是0x2d000.

* **执行软盘中的内容**

编写Haribote.nas文件如下, 结构很简单:

fin:

HLT

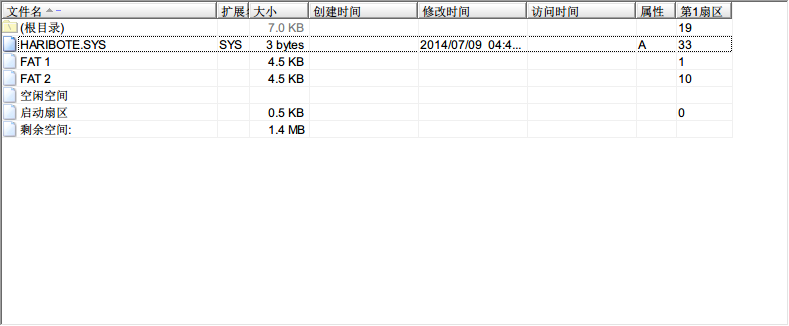
JMP fin

问题是如何把编译好的程序和IPL结合在一起?

我们可以先编译好IPL, 和之前一样做成一个img, 然后把程序写入img.

最直观的方法是把img文件写入U盘, 把程序写入U盘, 再把U盘打包成img,

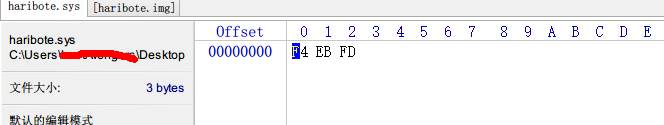
虽然有些迂回, 但是我们可以借助WinHex来直接写入, 用WinHex直接打开作者做好的Haribote.img ,可以发现它的内部格式如下:



发现Haribote.SYS处在0x4200的位置处 , 这是不是真的Haribote.sys?



也从Winhex里面打开Hraibote.sys:



只有一句, F4 EB FD, 和img文件0x4200处的完全一样, 和作者的总结一样:

向一个空软盘保存文件的时候,

1.文件名会写在0x2600以后的地方(这里我没有去验证);

2.文件的内容会写入到0x4200以后的地方.

非空软盘的情况没有考虑(这里开始我们就真正意义上使用了FAT12的文件系统了, 而不是自己设计一个文件系统).

知道了这个我们就可以知道怎么调用这个程序了, 使用作者提供的edimg.exe 可以方便地将程序写入到img中, 所以Makefile也要做相应的改动, 具体看Project文件夹day3里面的e文件夹.

这个Haribote.nas 所做的只是不断HLT, 我们并不知道这段代码是否真的执行, 作者这里使用了0x10来切换画面模式, 使得光标消失,

参数是AL = 0x13, AH = 0x00, 效果是屏幕全黑, 连光标也不能看到.

我这里采用的是0x10中断的另一个功能, 能够显示带颜色的字符串, 代码如下:

; Program

ORG 0xc200 ; 加载到 0x8200 + 0x4200 – 0x200(启动区没有被读入)

MOV AX,0

MOV ES,AX

MOV AX,loaded

MOV BP,AX ; ES:BP = 串地址

MOV CX,10 ; 串长度

MOV AX,0x1301 ;AH = 0x13，AL = 0x01

MOV BX,0x000b ; 页号BH = 0 黑底蓝字 BL = 0x0b

MOV DL,0

INT 0x10

JMP fin

loaded:

DB 0x0a, 0x0a

DB "Loaded."

DB 0x0a

DB 0

fin:

HLT

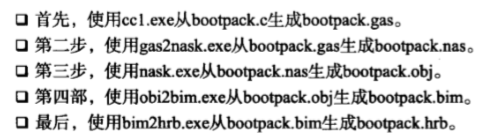
JMP fin

效果如下, 可以确定我们Haribote.sys里面的代码已经被执行.



* **进入32位模式&导入C语言\***

这里作者开始用我们看不懂的东西了, 给出了一个长长的asmhead.nas, 并且表示”先跳过这一部分”, 让我有点不爽, 但是毕竟不能半途而废不是? 这里的Makefile变得越来越复杂, 在这里我第一次碰到了困难, 由于粗心Makefile写少了几行, 导致编译莫名其妙地出错, 纠结了两天才发现, 抄代码都能抄错 = =, 废话略过, 作者还增加了一个Bootpack.c 大概就是我们以后的主战场了, 这里编译过程异常繁琐(作者啊你可不可以不要用你自己写的工具!? (╯‵□′)╯︵┻━┻),



以上所有的工具都是川和先生写的或者改写的(╯‵□′)╯︵┻━┻.

需要强调的是HariMain这个Main函数的函数名已经写死了, 不能够更换, 为了在C

语言里面使用HLT, (事实上现在C语言里面根本就没有函数可用), 增加naskfunc.nas.

所有文件均如下:

IPL.nas:

; hello-os

; TAB=4

; const

CYLS EQU 10

ORG 0x7c00

JMP entry

DB 0x90

DB "OS 0.01 "

DW 512

DB 1

DW 1

DB 2

DW 224

DW 2880

DB 0xf0

DW 9

DW 18

DW 2

DD 0

DD 2880

DB 0,0,0x29

DD 0xffffffff

DB "OS ver 0.01"

DB "FAT12 "

RESB 18

entry:

MOV AX,0

MOV SS,AX

MOV SP,0x7c00

MOV DS,AX

MOV AX,0x0820 ; 缓存位置 = ES:BX

MOV ES,AX

MOV CH,0 ; 柱面0

MOV DH,0 ; 磁头0

MOV CL,2 ; 扇区2

readloop:

MOV SI,0 ; 记录失败次数

retry:

MOV AH,0x02 ; 读盘

MOV AL,1 ; 一个扇区

MOV BX,0

MOV DL,0x00 ; 驱动器A:

INT 0x13 ; 调用磁盘中断

JNC next

ADD SI,1

CMP SI,5 ; 失败5次跳出

JAE error

MOV AH,0x00

MOV DL,0x00 ; 指定驱动器A:

INT 0x13 ; 重置并重试

JMP retry

next:

MOV AX,ES

ADD AX,0x0020

MOV ES,AX ; 段寄存器往后移0x0020 实际偏移 0x0020\*0x0010 = 0x0200 = 512d

ADD CL,1

CMP CL,18 ; 读到18扇区

JBE readloop; <= 则跳

MOV CL,1

ADD DH,1 ; 换一个磁头

CMP DH,2

JB readloop; <则跳

MOV DH,0

ADD CH,1 ; 柱面+1

CMP CH,CYLS

JB readloop

MOV [0x0ff0],CH

JMP 0xc200

error: ; 错误提示

MOV AX,0

MOV ES,AX

MOV AX,msg

MOV BP,AX ; ES:BP = 串地址

MOV CX,14 ; 串长度

MOV AX,0x1301 ;AH = 0x13，AL = 0x01

MOV BX,0x000c ; 页号BH = 0 黑底红字 BL = 0x0c

MOV DL,0

INT 0x10

fin:

HLT

JMP fin

msg:

DB 0x0a, 0x0a

DB "Load error."

DB 0x0a

DB 0

RESB 0x7dfe-$

DB 0x55, 0xaa

Asmhead.nas:

BOTPAK EQU 0x00280000

DSKCAC EQU 0x00100000

DSKCAC0 EQU 0x00008000

CYLS EQU 0x0ff0 ; 设定启动区

LEDS EQU 0x0ff1

VMODE EQU 0x0ff2 ; 关于颜色数目的信息，颜色的位数

SCRNX EQU 0x0ff4 ; 分辨率 X(Screen X)

SCRNY EQU 0x0ff6 ; 分辨率 Y(Screen Y)

VRAM EQU 0x0ff8 ; 图像缓冲区的起始地址

ORG 0xc200

MOV AL,0x13 ; VGA 显卡

MOV AH,0x00

INT 0x10

MOV BYTE [VMODE],8 ; 记录画面模式

MOV WORD [SCRNX],320

MOV WORD [SCRNY],200

MOV DWORD [VRAM],0x000a0000

; 取得键盘上的LED灯的状态

MOV AH,0x02

INT 0x16

MOV [LEDS],AL

MOV AL,0xff

OUT 0x21,AL

NOP

OUT 0xa1,AL

CLI

CALL waitkbdout

MOV AL,0xd1

OUT 0x64,AL

CALL waitkbdout

MOV AL,0xdf

OUT 0x60,AL

CALL waitkbdout

[INSTRSET "i486p"]

LGDT [GDTR0]

MOV EAX,CR0

AND EAX,0x7fffffff

OR EAX,0x00000001

MOV CR0,EAX

JMP pipelineflush

pipelineflush:

MOV AX,1\*8

MOV DS,AX

MOV ES,AX

MOV FS,AX

MOV GS,AX

MOV SS,AX

MOV ESI,bootpack

MOV EDI,BOTPAK

MOV ECX,512\*1024/4

CALL memcpy

MOV ESI,0x7c00

MOV EDI,DSKCAC

MOV ECX,512/4

CALL memcpy

MOV ESI,DSKCAC0+512

MOV EDI,DSKCAC+512

MOV ECX,0

MOV CL,BYTE [CYLS]

IMUL ECX,512\*18\*2/4

SUB ECX,512/4

CALL memcpy

MOV EBX,BOTPAK

MOV ECX,[EBX+16]

ADD ECX,3

SHR ECX,2

JZ skip

MOV ESI,[EBX+20]

ADD ESI,EBX

MOV EDI,[EBX+12]

CALL memcpy

skip:

MOV ESP,[EBX+12]

JMP DWORD 2\*8:0x0000001b

waitkbdout:

IN AL,0x64

AND AL,0x02

JNZ waitkbdout

RET

memcpy:

MOV EAX,[ESI]

ADD ESI,4

MOV [EDI],EAX

ADD EDI,4

SUB ECX,1

JNZ memcpy

RET

ALIGNB 16

GDT0:

RESB 8

DW 0xffff,0x0000,0x9200,0x00cf

DW 0xffff,0x0000,0x9a28,0x0047

DW 0

GDTR0:

DW 8\*3-1

DD GDT0

ALIGNB 16

bootpack:

Naskkfunc.nas:

;naskfunc

[FORMAT "WCOFF"] ; 制作目标文件的模式

[BITS 32] ; 制作32位模式用的机械语言

; 制作目标文件的信息

[FILE "naskfunc.nas"] ; 源文件名信息

GLOBAL \_io\_hlt ; 程序中包含的函数名

; 实际的函数

[SECTION .text] ;

\_io\_hlt:

HLT

RET

Bootpack.c:

void io\_hlt(void);

void HariMain(void)

{

fin:

io\_hlt();

goto fin;

}

万恶的Makefile:

TOOLPATH = ../z\_tools/

INCPATH = ../z\_tools/haribote/

MAKE = $(TOOLPATH)make.exe -r

NASK = $(TOOLPATH)nask.exe

CC1 = $(TOOLPATH)cc1.exe -I$(INCPATH) -Os -Wall -quiet

GAS2NASK = $(TOOLPATH)gas2nask.exe -a

OBJ2BIM = $(TOOLPATH)obj2bim.exe

BIM2HRB = $(TOOLPATH)bim2hrb.exe

RULEFILE = $(TOOLPATH)haribote/haribote.rul

EDIMG = $(TOOLPATH)edimg.exe

DEL = del

SHORTCUT = "D:\Program Files\Oracle\VirtualBox\VirtualBox.exe" --comment "OS1" --startvm "a5c4b0e6-e142-4720-98ee-056911204b29"

default :

$(MAKE) install

$(MAKE) run

$(MAKE) clean

ipl10.bin : ipl10.nas Makefile

$(NASK) ipl10.nas ipl10.bin ipl10.lst

asmhead.bin : asmhead.nas Makefile

$(NASK) asmhead.nas asmhead.bin asmhead.lst

bootpack.gas : bootpack.c Makefile

$(CC1) -o bootpack.gas bootpack.c

bootpack.nas : bootpack.gas Makefile

$(GAS2NASK) bootpack.gas bootpack.nas

bootpack.obj : bootpack.nas Makefile

$(NASK) bootpack.nas bootpack.obj bootpack.lst

naskfunc.obj : naskfunc.nas Makefile

$(NASK) naskfunc.nas naskfunc.obj naskfucn.lst

bootpack.bim : bootpack.obj naskfunc.obj Makefile

$(OBJ2BIM) @$(RULEFILE) out:bootpack.bim stack:3136k map:bootpack.map \

bootpack.obj naskfunc.obj

echo error

# 3MB+64KB=3136KB

bootpack.hrb : bootpack.bim Makefile

$(BIM2HRB) bootpack.bim bootpack.hrb 0

haribote.sys :asmhead.bin bootpack.hrb Makefile

copy /B asmhead.bin+bootpack.hrb haribote.sys

haribote.img : ipl10.bin haribote.sys Makefile

$(EDIMG) imgin:../z\_tools/fdimg0at.tek \

wbinimg src:ipl10.bin len:512 from:0 to:0 \

copy from:haribote.sys to:@: \

imgout:haribote.img

install:

$(MAKE) haribote.img

run :

echo Running...

$(SHORTCUT)

echo Finished

clean :

-$(DEL) \*.bin

-$(DEL) \*.lst

-$(DEL) \*.gas

-$(DEL) \*.obj

-$(DEL) bootpack.nas

-$(DEL) bootpack.map

-$(DEL) bootpack.bim

-$(DEL) bootpack.hrb

-$(DEL) haribote.sys

-$(DEL) \*.\*~

-$(DEL) \*~

echo Finished.

这次更新慢了, 下次争取早一点…