****

**《操作系统原理实验》**

**实验报告**

**（实验）**

FAT12文件仿真系统

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学院名称** | **：** | 数据科学与计算机学院 | | | | | |
| **专业（班级）** | **：** | 18级计算机科学与技术 | | | | | |
| **学生姓名** | **：** | 钟婕 | | | | | |
| **学号** |  | 18340225 | | | | | |
| **时间** | **：** | 2020 | 年 | 4 | 月 | 25 | 日 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | **成绩** | **:** |  |
| **实验** | **：** | **FAT12文件仿真系统** | | | |

* + - 1. **实验目的：**

1、满足疫情期间停课不停学，提供计算机相关专业大二学生学习操作系统相关知识的一个选择

2、提供一个综合编程实训项目，提高计算机相关专业大二学生的高级语言编程综合能力

3、提前学习操作系统原理课程的部分教学内容，减轻开学后的教学压力

4、探索网课这种新的教学形式，为学院正式推广网课提供参考。

* + - 1. **实验要求**

**在了解FAT12文件系统的结构与原理的基础上，**根据操作仿真的设计要求**，**在参考部分代码的辅助下，利用一种高级语言，设计并实现一个规定的文件操作和功能的仿真程序。**并**编写实验报告，描述实验工作的过程和必要的细节，如截屏或录屏。

三、**实验器材**

PC机一台，VMware Workstation，WinHex，Visual Studio2017。

* + - 1. **实验方案与过程**
* **实验原理：**

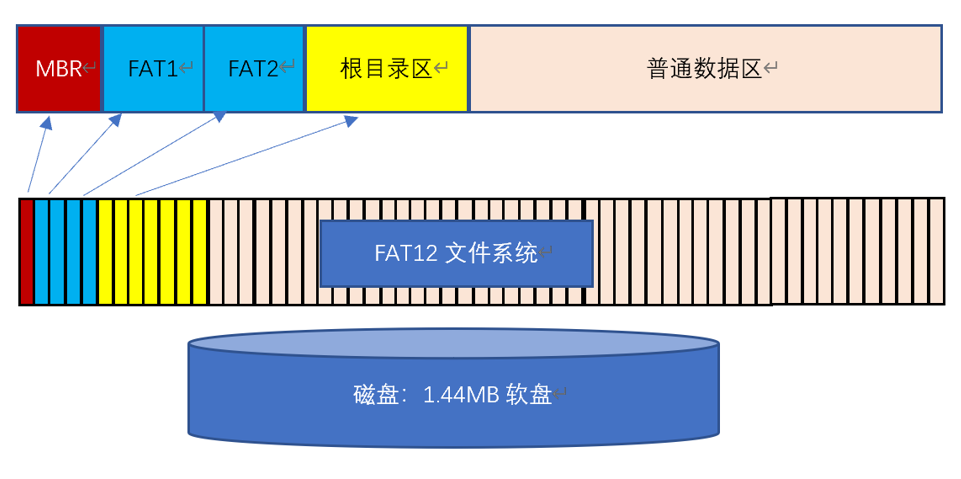
1. **FAT12文件系统的总体结构：**

本次实验要求的FAT12格式化的虚拟软盘大小为1.44MB。

具有以下关系：

1.44MB=1440KB=512B\*2880；

一个扇区的大小为512个字节，因此FAT12虚拟软盘共有2880个扇区，这2880个扇区根据FAT12文件系统的结构分配，具有以下的构成关系。



总共包含四个组成成分：

MBR（首扇区）包含引导扇区程序即主引导记录。

FAT表（9+9个扇区）包含FAT1和FAT2两个表，表2是表一的拷贝。FAT表记录着文件数据簇的链接关系，每12位为一个簇，记录着下一簇的簇号，簇号对应着数据区的数据块。

根目录区（14个扇区）包含着根目录下的各个目录项信息，每个目录项占32字节，可以存储224个目录项。

普通数据区（2847个扇区）包含着文件内容以及子目录文件的相关信息。

**2、首扇区的512字节分配。**

已知首扇区为512字节，包含引导扇区程序，但引导扇区程序并不占据这512个字节的全部，在这512个字节中描述了FAT12文件系统的相关性质，具体如下。



从上表可知，FAT12文件系统相关参数与格式均包含在首扇区中，因此我们可以从首扇区的特定位置提取相关FAT12格式的信息。便于我们对FAT12文件系统的格式进行检查。

**3、FAT表的结构**

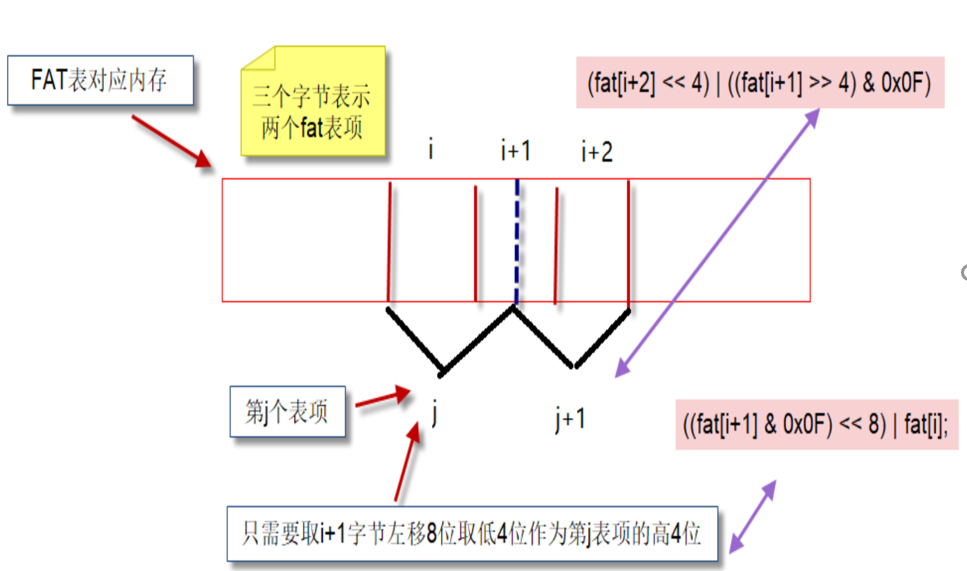
FAT表中的FAT1、FAT2是相互备份的关系，两者内容完全一样。其实，FAT表就是一个关系图，记录了文件数据的链接先后关系。FAT12文件系统中的“12”就体现在

FAT表中每个表项占12位，而且FAT表规定前两个表项不使用（分别作为坏簇和文件结束标志）。

那么FAT表中的每个表项是如何体现文件数据的先后链接关系呢？我们可以把FAT表中的每一个表项看成一个链接指针，一个文件数据的内容就是由一串FAT表项对应的数据块串接起来的。文件的链接关系为：第一个数据块的块号即为首簇号，第二个数据块的块号存放在第一簇对应的链接指针中，第三个数据块的块号存放在第二簇对应的链接指针中，以此类推，最后一个簇存放的是文件结束标志：0xFFF。

除了知道文件链接关系，由于FAT12文件系统的一个文件其实就是一个隐式链表，文件链接关系在FAT表中，而文件数据块在普通数据区中，因此需要掌握FAT簇号与数据块逻辑扇区号的对应关系：最小的有效簇号为2对应第一个数据块的逻辑扇区号33——>数据块逻辑扇区号=簇号+31.

另外，由于每个FAT表项占12位，因此三个字节大小包含了两个簇，关于簇号的提取方法如下：



**4、根目录区的结构**

根目录区主要是存放根目录下的若干个目录项的区域，每个目录项占32字节，根目录区总共有14个扇区，每个扇区512个字节，因此根目录区总共可以存放224个目录项。

每个目录项代表根目录中的一个文件索引，目录项的结构如下：

DIR\_Name:文件名8字节，扩展名3字节

DIR\_Attr：文件属性

Reserve：保留位

32 Bytes DIR\_WrtTime：最后一次写入时间

DIR\_WrtDate：最后一次写入日期

DIR\_FstClus：文件开始的簇号

DIR\_FileSize：文件大小

其中：

文件属性中：27H、20H为普通文件；10H为目录文件

文件开始的簇号对于显示文件内容，找到相应数据块很有帮助。

**5、普通数据区**

myName、myNumber为声明与初始化字符串伪指令的标志，代表着相对于程序开始处的偏移地址，也是相对于段基地址的偏移地址，因此将它赋值给BP寄存器，CX寄存器存储字符串字节数，AH为13H代表功能号，AL为光变位置，BH为显示的页号、BL为显示属性，此处为蓝底浅品红无闪烁，DX为行号与列号位置。INT 10H调用中断号为10H的中断服务程序。

1. **显示字符运动轨迹**

为了在屏幕上显示出字符运动的效果，每次运动需要有一定的时间间隔，因此定义计时器数据常量，控制画框运动速度。

delay equ 50000 ;计时器延迟计数;用于控制画框的速度

ddelay equ 580 ;计时器延迟计数;用于控制画框的速度

由于字符是不断在运动的，因此每次时间延迟的计算可以用一个循环实现，间隔指定的计数时间（50000\*580），字符朝着一定方向运动一个单位.

Loop段即为运动间隔的计算过程，间隔之后，让字符在相应运动方向运动。

Loop:

dec word[num] ；递减计数变量，产生延迟一共50000\*280

jnz Loop ；大于0时，跳转

mov word[num],delay

dec word[dnum]

jnz Loop ；大于0时，跳转

mov word[num],delay

mov word[dnum],ddelay

mov al,1

cmp al,byte[initial] ；类似于switch语句，朝相应方向运动

jz DnRt

mov al,2

cmp al,byte[initial]

jz UpRt

mov al,3

cmp al,byte[initial]

jz UpLt

mov al,4

cmp al,byte[initial]

jz DnLt

jmp $

通过递减计数变量，间隔延迟50000\*280，利用类似于switch语句来判断字符的运动方向，跳转到相应方向代码段执行字符运动。initial为声明初始化运动方向的标志，初始声明为Dn\_Rt(右下)方向，每次运动之后判断是否更新initial内容（即是否发生反射，改变运动方向）。

下选取四个运动方向中的其中一个DnRt(右下方向)说明运动过程的程序设计，包括遇到边界反弹的情况。

不妨假设字符A从（x,y）开始向右下方运动，那么下一时刻的位置确定，将分为三种情况：字符未碰触边界、字符碰触行边界、字符碰触列边界。这里，行号的范围为0~24，列号的范围为0~79。显而易见，当字符未碰触边界时，下一时刻的字符位置应该为（x+1，y+1）。如果字符碰触到行边界，即x+1后与25相等，那么将发生反弹，字符的运动方向将变为右上方向，即程序跳转至dr2ur段执行。如果如果字符碰触到列边界，即y+1后与80相等，那么将发生反弹，字符的运动方向将变为左下方向，即程序跳转至dr2dl段执行。

而在发生反弹变换方向的代码中，只需要将到达边界的相应参数重置，同时更改initial中的内容即可。例如，在dr2ur段，字符运动到行边界，因此将x行坐标置为24-1=23即可，y列坐标不变，而将initial的内容重置为右上方向。而在dr2dl段，字符运动到列边界，因此将y行坐标置为79-1=78即可，x行坐标不变，而将initial的内容重置为左下方向。

具体代码如下所示：

dr2dl:

mov word[y],78 ;将y列坐标置为79

mov byte[initial],Dn\_Lt ;将运动方向更改为左下

jmp display ;显示字符

dr2ur:

mov word[x],23 ;将x行坐标置为23

mov byte[initial],Up\_Rt ;将运动方向更改为右上

jmp display ;显示字符

DnRt:

inc word[x] ;坐标x行加一

inc word[y] ;坐标y列加一

mov bx,word[x]

mov ax,25

sub ax,bx ;比较行数是否到达边界

jz dr2ur ;到达则由右下方向变为右上

mov bx,word[y]

mov ax,80

sub ax,bx ;比较列数是否到达边界

jz dr2dl ;到达则由右下方向变为左下

jmp display ;未到达边界则显示字符

确定了字符下一时刻的位置以及运动方向之后，则需要显示字符，在本次实验中即为display段中的内容。display段代码向内存中某个地址空间写入需要显示的字符，则写入的内容将立即以给定的显示属性出现在屏幕中。我们需要根据当前字符的行号和列号计算出预期写入的显存地址。由于行号的范围为0~24、列号的范围为0~79，我们可以通过公式：（行号×80+列号）×2得到预期的显存偏移地址。在确定了字符的显示属性之后，我们便可向计算得出的显存偏移地址与段基地址B800H组合的串地址空间写入相应内容。

在此处，为了凸显个性化的设计，我设计字符运动时字符为26个大写字母依次循环显示，而字符的显示属性也是从01H到07H依次改变。color为声明初始化字符显示属性的标志，初始化为01H；char为声明初始化字符的标志，初始化为‘A’，两者均为字节大小。

在每次显示时，写入的串地址为字大小的[gs:bp]的内存空间，ax寄存器高位ah存放显示属性[color]，低位al存放显示的字符[char]。

每次显示完成之后，将color、char的内容递增，递增后判断color的内容是否与08H相等，如果相等则跳转到circle1重置color的内容为01H；同样，判断char的内容时候与90相等，如果相等则跳转到circle2重置char的内容为65。这样即可实现个性化的设计。

具体代码如下：

display:

mov ax,word[x]

mov bx,80

mul bx

add ax,word[y]

mov bx,2

mul bx ;计算显存偏移地址

mov bp,ax

mov ah,[color] ;显示属性

mov al,byte[char] ;显示字符

mov word[gs:bp],ax ;向相应显存地址写入

add byte[color],1

add byte[char],1 ;递增

mov bl,8

cmp byte[color],bl ;判断是否相等

jz circle1 ;相等则跳转到circle1

mov bl,90

cmp byte[char],bl ;判断是否相等

jz circle2 ;相等则跳转到circle2

jmp Loop

circle1:

mov byte[color],1 ;重置color的内容

mov bl,90

cmp byte[char],bl

jz circle2

jmp Loop

circle2:

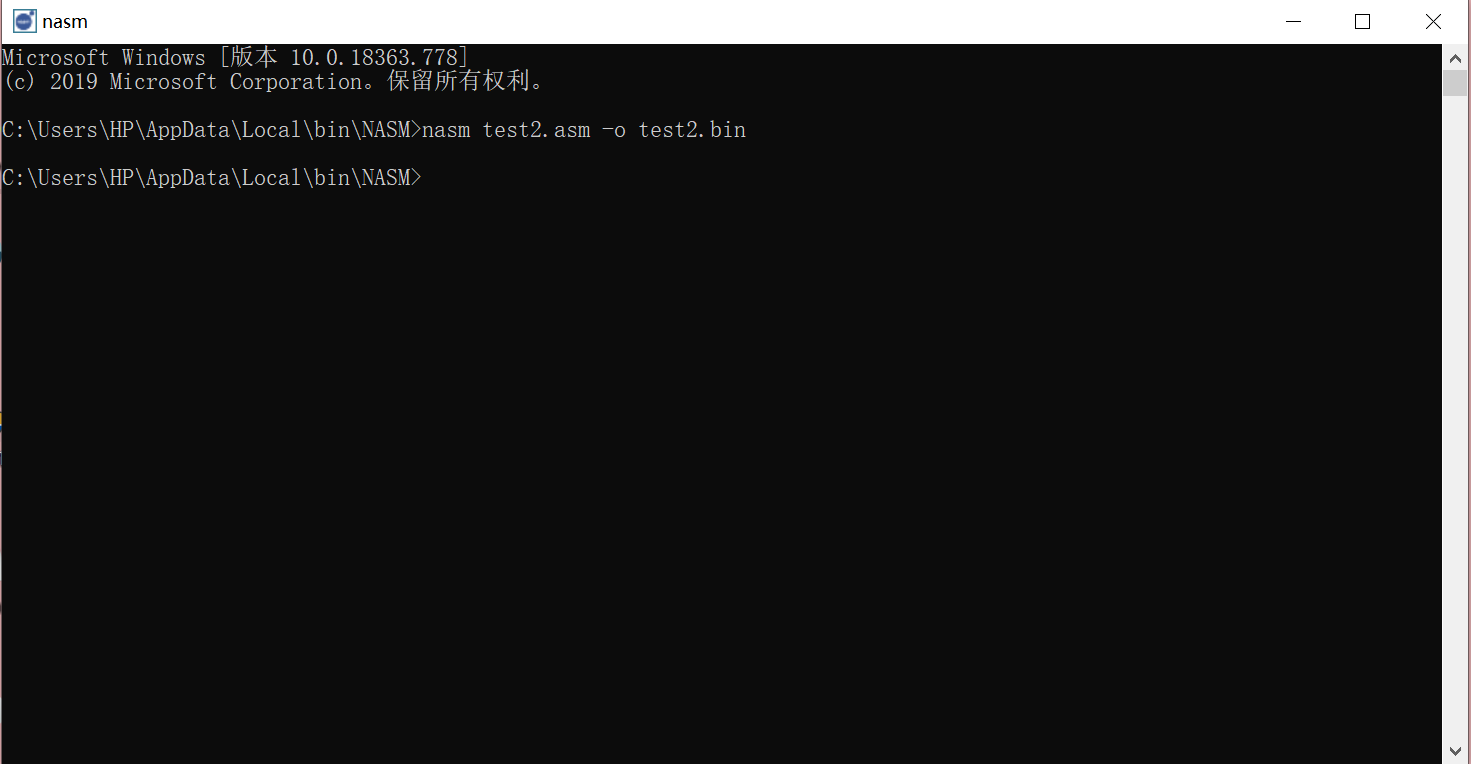
mov byte[char],65 ;重置char的内容

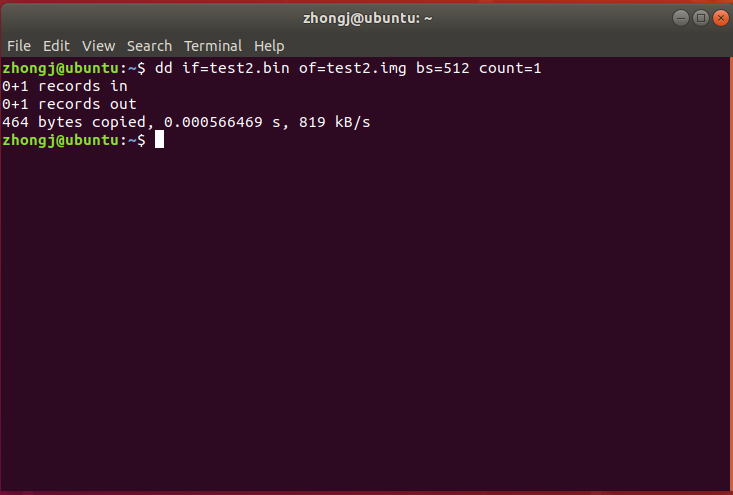
jmp Loop

**五、实验结果**

将上述包含上述代码的“test2.asm”文件利用NASM汇编生成“test2.bin”二进制文件，并在虚拟机中利用dd命令生成“test2.img”软盘映像文件。

操作过程如下所示：





这条dd命令能够创建一个大小为512字节，名字为test2.img，写入了test2.bin的虚拟软盘映像，其中test2.bin为读取位置，test2.img为写入位置，bs=512是每次读写大小，count=1指的是只读取一次。

基于上述操作就可以得到一个扇区（512B）大小的虚拟软盘映像，根据实验要求需要将该程序机器码放入1.44MB的虚拟软盘中，即创建一个包含此引导扇区的1.44MB虚拟软盘映像。如下：

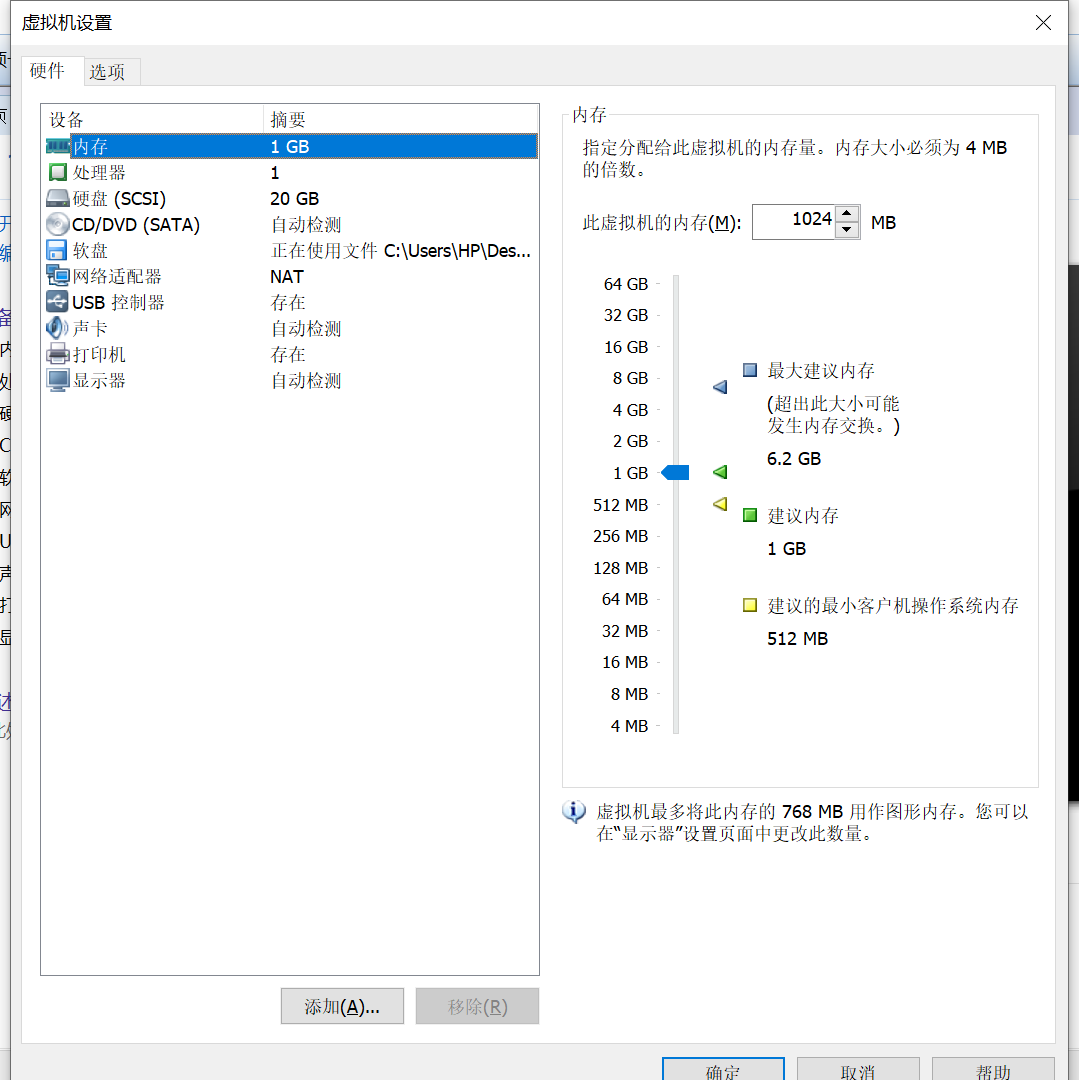
手机屏幕的截图

描述已自动生成

首先我们先创建一个空的大小为1.44MB的虚拟软盘映像empty.img。

然后将empty.img第一个扇区之后的数据拷贝到我们之前制作的包含主引导记录的镜像文件test2.img后。那么test2.img即为大小为1.44MB的包含主引导记录的虚拟软盘镜像文件。

接着我们创建虚拟机，完成虚拟机的基本配置：



配置完成后，开启此虚拟机，虚拟机上程序运行结果如下：

图片包含 人, 室内, 一群, 照片

描述已自动生成

显而易见，程序运行结果与实验要求相符合。

六、实验总结：

七、参考文献：