

# 实验一

## 计算机科学与技术（大数据）专业

17 级计科 姜洋帆

学号：17341068

### 一、实验题目： 接管裸机的控制权

### 二、实验目的：

1. 配置实验环境
2. 体验裸机编程

### 三、实验要求：

1. 搭建和应用实验环境
2. 接管裸机的控制权

### 四、实验方案：

#### 1.实验环境

采用 virtual box 虚拟机作为实验平台，实验工具的组合为 GCC+NASM。

其他工具：

notepad++， sublime 以及自己编写的一个 C 程序。

自己编写的 C 程序用于将二进制流文件转换为软盘映像文件（.img 格式），并将软盘大小扩展为 1.44M，Notepad++用于编辑汇编代码，sublime 查看和编辑二进制流文件（以及软盘映像文件）

#### 2.程序功能及设计

首先将 virtual box 初始化一个实验用的虚拟机，内存配置为 16M，硬盘配置为 16M（实际上从软盘启动，这个硬盘在这个实验没有作用）。

用测试虚拟机可以正常工作后（显示一个字符），开始编写程序汇编代码，做出的主要调整有：

1. 改变字符的颜色，实现思路为将 ah 寄存器的赋值放在 start 模块中初始化，之后每次执行 show 模块时，add ah,1 来改变字符的颜色属性。
2. 控制字符的长度。按照老师给的代码来执行，由于没有擦除之前显示的字符，字符会一直显示在屏幕上直至占满所有空间。通过添加一些变量来记录字符前几个状态的坐标，在每次更新时擦除最后一个坐标的显示，可以做到控制字符显示的长度，

避免一直显示在屏幕上（类似于贪吃蛇）。

### 3. 在中间显示姓名学号

另外，为了防止移动的字符在经过显示学号姓名的区域时，把原本的信息覆盖，需要不断调用显示姓名学号的模块，以保证姓名学号会一直显示在屏幕上，不会被覆盖。个人的做法是在每次调用显示移动字符 A 的功能后，就调用一次显示字符串的功能。

#### 关键代码如下：

数据定义部分，增加一些坐标的变量

```
datadef:
    count dw delay
    dcount dw ddelay
    rdul db Dn_Rt          ; 向右下运动
    x     dw 7
    y     dw 0
    x2    dw 7
    y2    dw 0
    x3    dw 7
    y3    dw 0
    x4    dw 7
    y4    dw 0
    x5    dw 7
    y5    dw 0
    char db 'A'
```

更新记录的坐标，将最后一个记录的坐标显示为黑色，达到消除字符，控制屏幕上字符数量的效果

```
Erase:          ;erase the tail of the string
    mov ax,word[x4]
    mov word[x5],ax
    mov ax,word[y4]
    mov word[y5],ax
    mov ax,word[x3]
    mov word[x4],ax
    mov ax,word[y3]
    mov word[y4],ax
    mov ax,word[x2]
    mov word[x3],ax
    mov ax,word[y2]
    mov word[y3],ax
    mov ax,word[x]
    mov word[x2],ax
    mov ax,word[y]
    mov word[y2],ax

;xor ax,ax          ; 计算显存地址
    mov ax,word[x5]
    mov bx,80
    mul bx
    add ax,word[y5]
    mov bx,2
    mul bx
    mov bp,ax
    mov ah,00h          ;闪烁、背景 RGB、高亮、字体 RGB 0000 0000
    mov al,byte[char]   ;AL(ax 低位)显示字符值
    mov word[gs:bp],ax   ;将字符值和颜色送到要显示字符的显存地址
    ret
```

显示字符的函数，这里将 **ah** 每次递增 **1**，达到改变颜色的效果。

```
show:
;   xor ax,ax           ; 计算显存地址
call DisplayStr
    mov ax,word[x]
mov bx,80
mul bx
add ax,word[y]
mov bx,2
mul bx
mov bp,ax
add ah,1
mov al,byte[char]       ;AL(ax 低位)显示字符值
mov word[gs:bp],ax      ;将字符值和颜色送到要显示字符的显存地址
jmp loop1
```

显示字符串：

设置好串长度以及颜色信息，调用 **10** 号中断即可

```
DisplayStr:
mov bp, BootMessage
;mov bp,ax      ;ES:BP = 串地址
mov cx,12 ;CX=串长度
mov ax,01301h ;AH = 13, AL = 01h
mov bx,000ch
mov dl,30
mov dh,10
int 10h
ret
```

## C 程序源代码：

效果为将编译生成的二进制文件转为 **.img** 格式文件，同时用 **0** 填充至 **1.44M** 字节。软盘大小必须为 **1440\*1024**，否则用 **virtual box** 加载软盘时，会出现如下错误



```

#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#define BUF_SIZE 20
char buf[BUF_SIZE];
char fileName[BUF_SIZE];

void input();

char scrfilepath[100] = "D:\\Study\\大二下\\Operation System\\Exp#2\\";
char destfname[100] = "D:\\Study\\大二下\\Operation System\\Exp#2\\";
FILE *scrFile;
FILE *destFile;

int main()
{
    input();
    getchar();
    getchar();
    return 0;
}

void input() {
    char filePath[150];

    printf("output file name:");
    scanf_s("%s", fileName, 20);
    strcpy_s(filePath, 100, destfname);
    strcat_s(filePath, fileName);
    fopen_s(&destFile, filePath, "wb");
    if (destFile == NULL) {
        printf("can't find file to write");
        getchar();
        getchar();
        exit(1);
    }

    printf("bin file name:");
    scanf_s("%s", fileName, 20);

    strcpy_s(filePath, 100, scrfilepath);
    strcat_s(filePath, fileName);
    fopen_s(&scrFile, filePath, "rb");
    if (scrFile == NULL) {
        printf("can't find file!\n");
        getchar();
        getchar();
        exit(1);
    }

    // get file size
    fseek(scrFile, 0L, SEEK_END);
    long fileSize = ftell(scrFile);
    long toAdd = 1440 * 1024 - fileSize;
    //1440*1024
    fclose(scrFile);
}

```

```

fopen_s(&scrFile, filePath, "rb");
printf("%d\n", fileSize);
//write file content
int len = 0;
while ((len = fread(buf, 1, BUF_SIZE, scrFile)) >= BUF_SIZE) {
    fwrite(buf, 1, BUF_SIZE, destFile);
}
fwrite(buf, 1, len, destFile);
//printf("%d\n", len);

bool a = 0;
//a[0] = 0;
while (toAdd > 0) {
    fwrite(&a, 1, 1, destFile);
    toAdd--;
}

fclose(scrFile);
fclose(destFile);
}

```

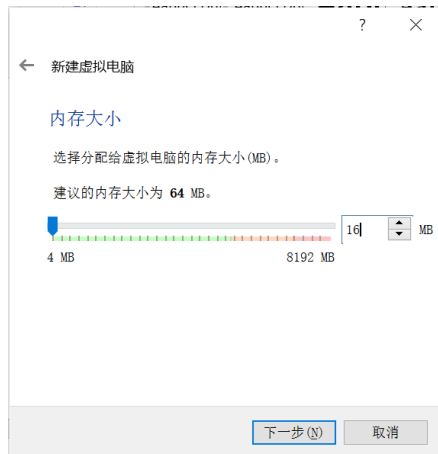
## 五、实验过程：

### 1.配置实验环境

使用 virtual box 配置实验环境，操作如下

新建->类型/版本选择 Other->分配内存和磁盘大小





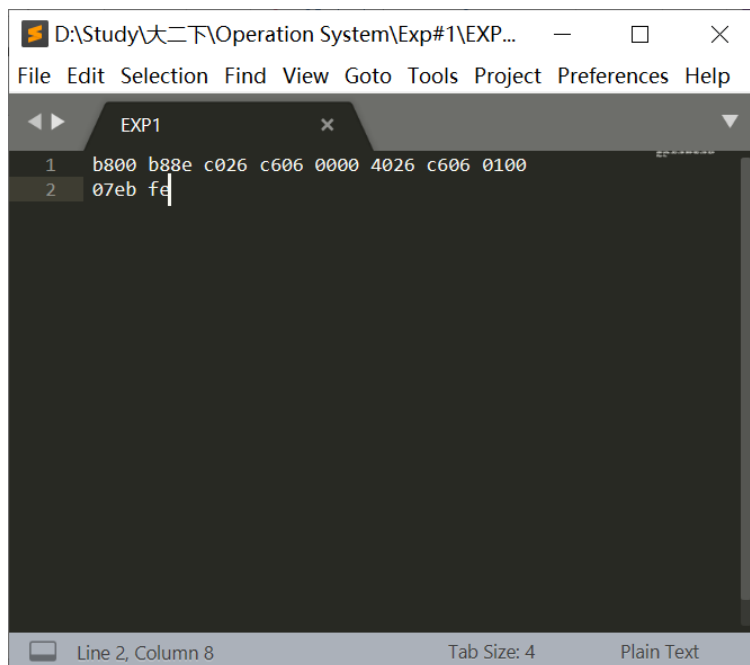
下面来尝试生成第一个软盘映像文件。采用 PPT 里在裸机显示@字符的例子。ppt 中的代码有些错误，需要设置地址 `org 07c00h`，使得程序加载到内存地址 `07c00`

编译：

```
D:\Study\大二下\Operation System\Exp#1>nasm EXP1.asm
D:\Study\大二下\Operation System\Exp#1>
```

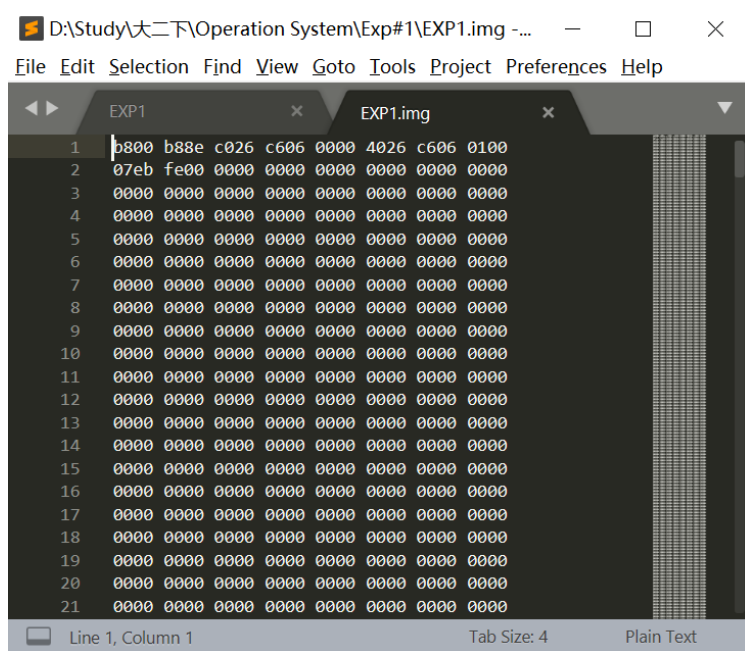
生成机器码

|      |                 |    |      |
|------|-----------------|----|------|
| EXP1 | 2019/3/16 18:02 | 文件 | 1 KB |
|------|-----------------|----|------|

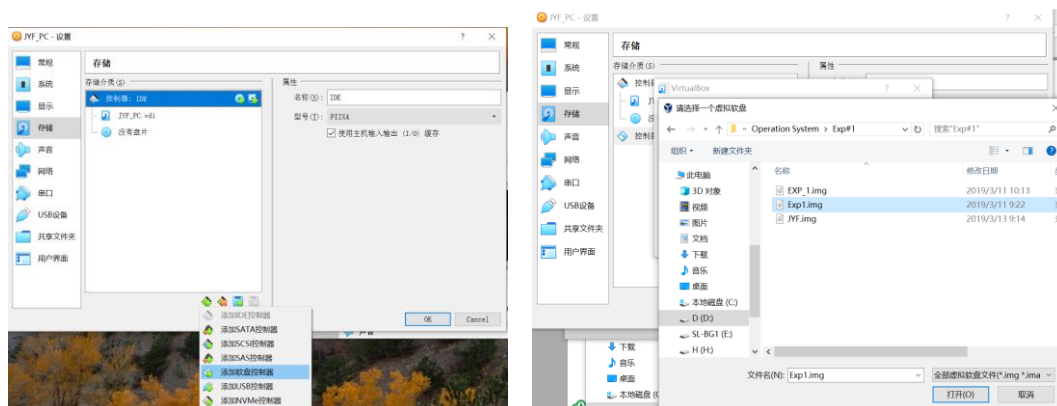


用 C 语言编写的程序将文件扩展为 1.44M 的 img 格式文件

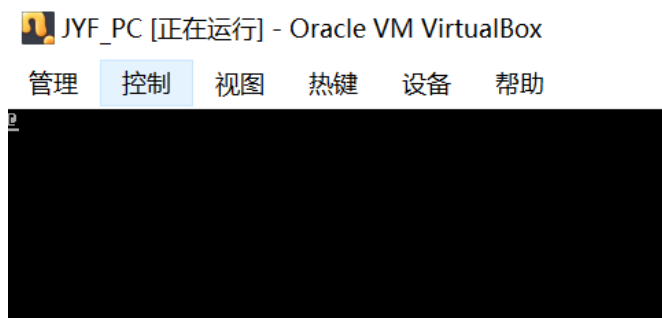
|          |                 |        |          |
|----------|-----------------|--------|----------|
| EXP1.img | 2019/3/16 23:39 | 光盘映像文件 | 1,440 KB |
|----------|-----------------|--------|----------|



选择添加软盘控制器，将生成的 img 文件作为引导



可以看到配置的虚拟机裸机左上方显示出一个 '@' 字符，表明环境已经配置成功了



接下来开始正式做实验

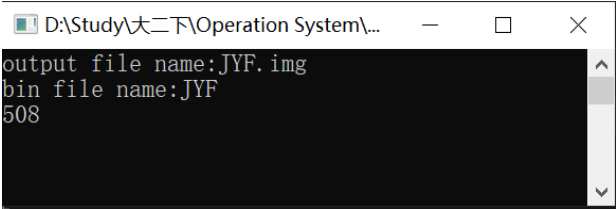
首先按照“实验方案”部分的方法，修改老师给的汇编源码，主要目的是实现两个功能：

1. 修改字符颜色
2. 控制字符长度

具体代码见附件，代码关键部分已贴在“实验方案”部分。  
使用 nasm 汇编




```
D:\Study\大二下\Operation System\Exp#1>nasm JYF.asm
D:\Study\大二下\Operation System\Exp#1>
```

将文件扩展至 1.44M

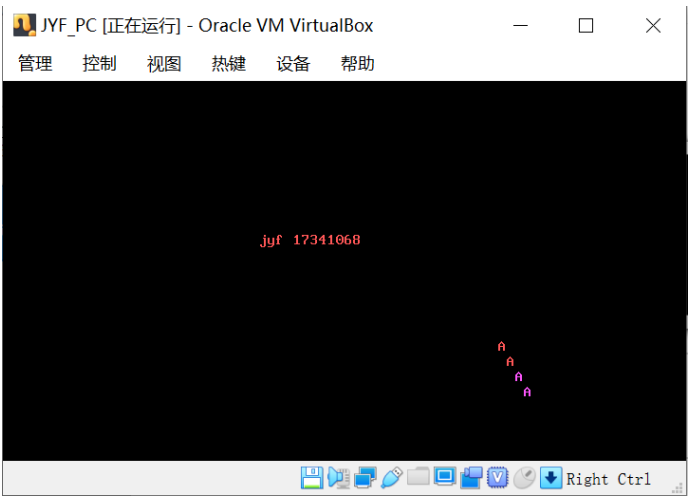


```
output file name:JYF.img
bin file name:JYF
508
```

返回的数字为二进制文件的大小（字节），需要确保小于等于 510 字节，511，512 两个字节为标志位 55aa。

|   |         |                 |                  |          |
|---|---------|-----------------|------------------|----------|
|  | JYF     | 2019/3/16 23:46 | 文件               | 1 KB     |
|  | JYF.asm | 2019/3/14 10:16 | Assembler Source | 5 KB     |
|  | JYF.img | 2019/3/16 23:47 | 光盘映像文件           | 1,440 KB |

在 virtual box 设置中将 JYF.img 设置为控制器，然后运行虚拟机，查看运行结果，实验结果截图如下：







## 六、实验总结

这是操作系统的第一个实验，算是一个入门的小实验吧，主要的目的就是配置好实验环境，了解计算机启动的相关原理，内容不算太难。

由于之前有使用虚拟机的经历，搭建实验环境部分没有遇到什么问题，实验主要的难点集中在了汇编代码上。

实验采用 x86 汇编，由于之前没有学习过相关的汇编语言知识，在学习 x86 汇编语法上稍微花了一些时间，好在老师给出了示例程序，已经完成了绝大部分工作（字符串移动、反弹等功能），自己实现的实际上只是在屏幕中间显示个人信息的字符串，几行简单的代码就可以解决。另外实现的功能一个是变色，改变 ah 寄存器的值即可；而另一个功能，控制字符长度就相对麻烦一下，需要定义额外的一些变量，记录上几个状态下字符的坐标信息，然后每更新一次字符时，就要同时对所有这些变量同步更新，并擦除最后一个变量。

这里有一个小问题需要注意，就是文件编译后大小不能超过 512 字节（一个扇区），否则程序无法被全部加载进内存运行，会出现各种问题。同时字符串大小也不能设错（CX 寄存器）。在做实验的过程中，还发现在 virtual box 环境下，即使软盘第 511, 512 字节没有设置为标记 55aa，也能够正常运行。

整个实验下来的感受就是，汇编代码不方便调试，每次更改代码都要重新编译生成软盘映像文件，在跑虚拟机来看效果，之后可能需要找一些方便的开发环境或者调试工具，提高汇编代码调试的效率。