中山大學
SUN YAT-SEN UNIVERSITY

《操作系统实验》

实验报告

（实验二）

学 院 名 称 ：	数据科学与计算机学院
专业（班级）：	16 计科 2 班
学 生 姓 名 ：	杨志成
学　　号 ：	16337281
时　　间 ：	2018　年　3　月　17　日

目录

成 绩 ：

实验一 ：加载用户程序的监控程序

一. 实验目的

(1) 掌握基础汇编语言，并灵活运用；
(2) 掌握基本监控程序的运行
(3) 掌握扇区与内存之间的关系
(4) 学会运用基础汇编语言编写稍复杂的
(5) 掌握基本中断指令如 INT 16H，INT 13H

二. 实验要求

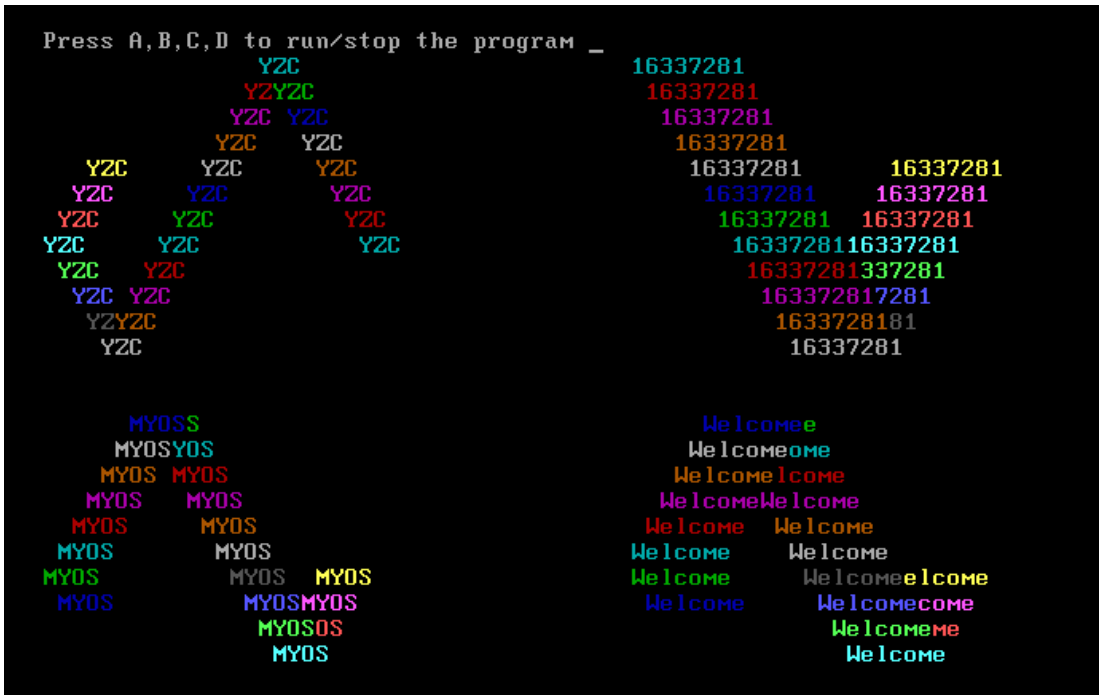
（1）设计四个（或更多）有输出的用户可执行程序，分别在屏幕1/4区域动态输出字符，如将用字符‘A’从屏幕左边某行位置45度角下斜射出，保持一个可观察的适当速度直线运动，碰到屏幕相应1/4区域的边后产生反射，改变方向运动，如此类推，不断运动；在此基础上，增加你的个性扩展，如同时控制两个运动的轨迹，或炫酷动态变色，个性画面，如此等等，自由不限。还要在屏幕某个区域特别的方式显示你的学号姓名等个人信息。

（2）修改参考原型代码，允许键盘输入，用于指定运行这四个有输出的用户可执行程序之一，要确保系统执行代码不超过512字节，以便放在引导扇区

（3）自行组织映像盘的空间存放四个用户可执行程序

三、实验方案

本次试验中，我同实验1一样，采取了循序渐进的思路，先实现了比较简单的版本，然后慢慢增加程序的功能，最后达到实验目的。一开始我只实现了监控程序引导一个子程序运行，接下来我成功引导了四个子程序的运行，最后我成功通过分时操作让四个子程序一起运行了起来,效果如下 ：



（1） 基础原理：

BIOS调用

BIOS是英文"Basic Input Output System"的缩略语，直译过来后中文名称就是"基本输入输出系统"。其实，它是一组固化到计算机内主板上一个ROM芯片上的程序，它保存着计算机最重要的基本输入输出的程序、系统设置信息、开机后自检程序和系统自启动程序。 其主要功能是为计算机提供最底层的、最直接的硬件设置和控制。

BIOS芯片中主要存放：

a)自诊断程序：通过读取CMOSRAM中的内容识别硬件配置，并对其进行自检和初始化；

b)CMOS设置程序：引导过程中，用特殊热键启动，进行设置后，存入CMOS RAM中；

c)系统自举装载程序：在自检成功后将磁盘相对0道0扇区上的引导程序装入内存，让其运行以装入DOS系统；

d)主要I/O设备的驱动程序和中断服务：由于BIOS直接和系统硬件资源打交道，因此总是针对某一类型的硬件系统，而各种硬件系统又各有不同，所以存在各种不同种类的BIOS，随着硬件技术的发展，同一种BIOS也先后出现了不同的版本，新版本的BIOS比起老版本来说，功能更强

BIOS中中断例程即BIOS中断服务程序

a) 是微机系统软、硬件之间的一个可编程接口，用于程序软件功能与微机硬件实现的衔接。DOS/Windows操作系统对软、硬盘、光驱与键盘、显示器等外围设备的管理即建立在系统BIOS的基础上。程序员也可以通过 对INT 5、INT 13等终端的访问直接调用BIOS终端例程。

调用BIOS中断服务程序的方法

b) 每个中断服务有特定的参数，一般使用指定的寄存器传递参数；

c) 利用软中断指令调用

d) BIOS中断调用的一般格式为：

mov ah,功能号

……；设置各种入口参数

int 中断号

本次实验用到的中断控制指令：

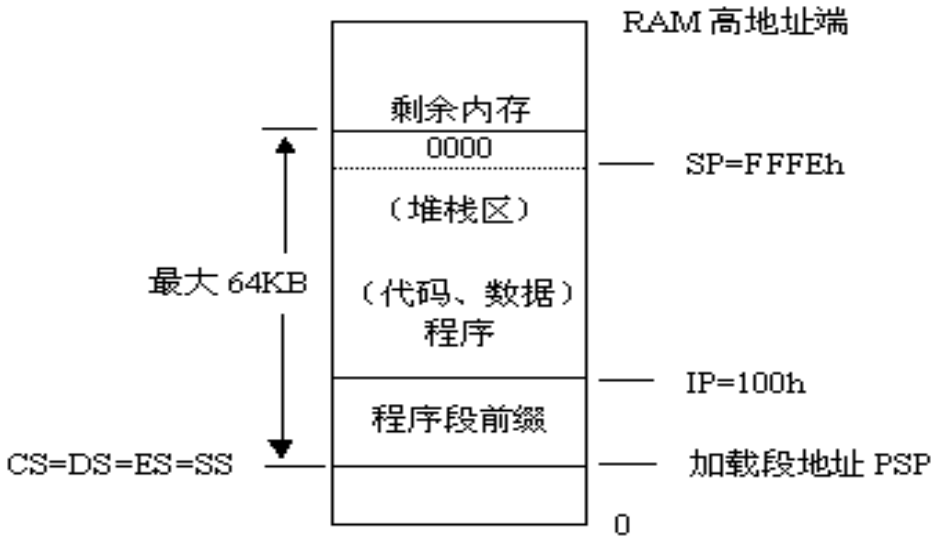
功能	**中断号**	**功 能 号**
插入空行上滚显示页窗口	10H	06H
以电传方式显示单个字符	10H	0EH
显示字符串	10H	13H
复位磁盘系统	13H	00H
读扇区	13H	02H
读下一个按键	16H	00H

当我们需要读入软盘扇区程序时，需运用int 13h

读扇区	13H	02H	AL：扇区数(1~255) DL：驱动器号(0 和 1 表示软盘，80H 和 81H 等表示硬盘或 U 盘) DH：磁头号(0~15) CH：柱面号的低 8 位 CL：0~5 位为起始扇区号(1~63)，6~7 位为硬盘柱面号的高 2 位(总共 10 位柱面号，取值 0~1023) ES：BX：读入数据在内存中的存储地址	返回值： ■ 操作完成后 ES：BX 指向数据区域的起始地址 ■ 出错时置进位标志 CF=1，错误代码存放在寄存器 AH 中 ■ 成功时 CF=0、AL=0

.com文件格式：

- COM（command file，命令文件）是CP/M和DOS的一种原始二进制可执行格式，以.com为扩展名。COM文件非常简单，没有文件头、没有元数据，只有代码和数据。
- COM文件会被装载到当前段的0x100（256）处，不能重新定位。由于不能分段，所以COM文件的大小必须≤64KB-256B，且不能有独立的数据段和堆栈段，程序的所有代码和数据都必须位于一个段中。
- 另外，在Windows操作系统的64位版本中，不再支持COM程序的运行。
- DOS加载COM程序的内存映象图



补充：有关批处理系统与监控程序

- 批处理系统

又名批处理操作系统。批处理是指用户将一批作业提交给操作系统后就不再干预，由操作系统控制它们自动运行。这种采用批量处理作业技术的操作系统称为批处理操作系统。批处理操作系统分为单道批处理系统和多道批处理系统。批处理操作系统不具有交互性，它是为了提高CPU的利用率而提出的一种操作系统。

- 监控程序是操作系统的最早期的形式
  - 获取计算机硬件系统的控制权
  - 提供计算机输入设备和输出的控制程序
  - 控制用户程序的执行
- 如果我们要为IBM_PC开发监控程序级的操作系统，那么应该怎样完成任务?
  - 因为该机器的BIOS能够控制输入设备和输出设备，所以我们的工作主要实现控制用户程序的执行这一项。

（2）实验工具和环境

实验支撑环境

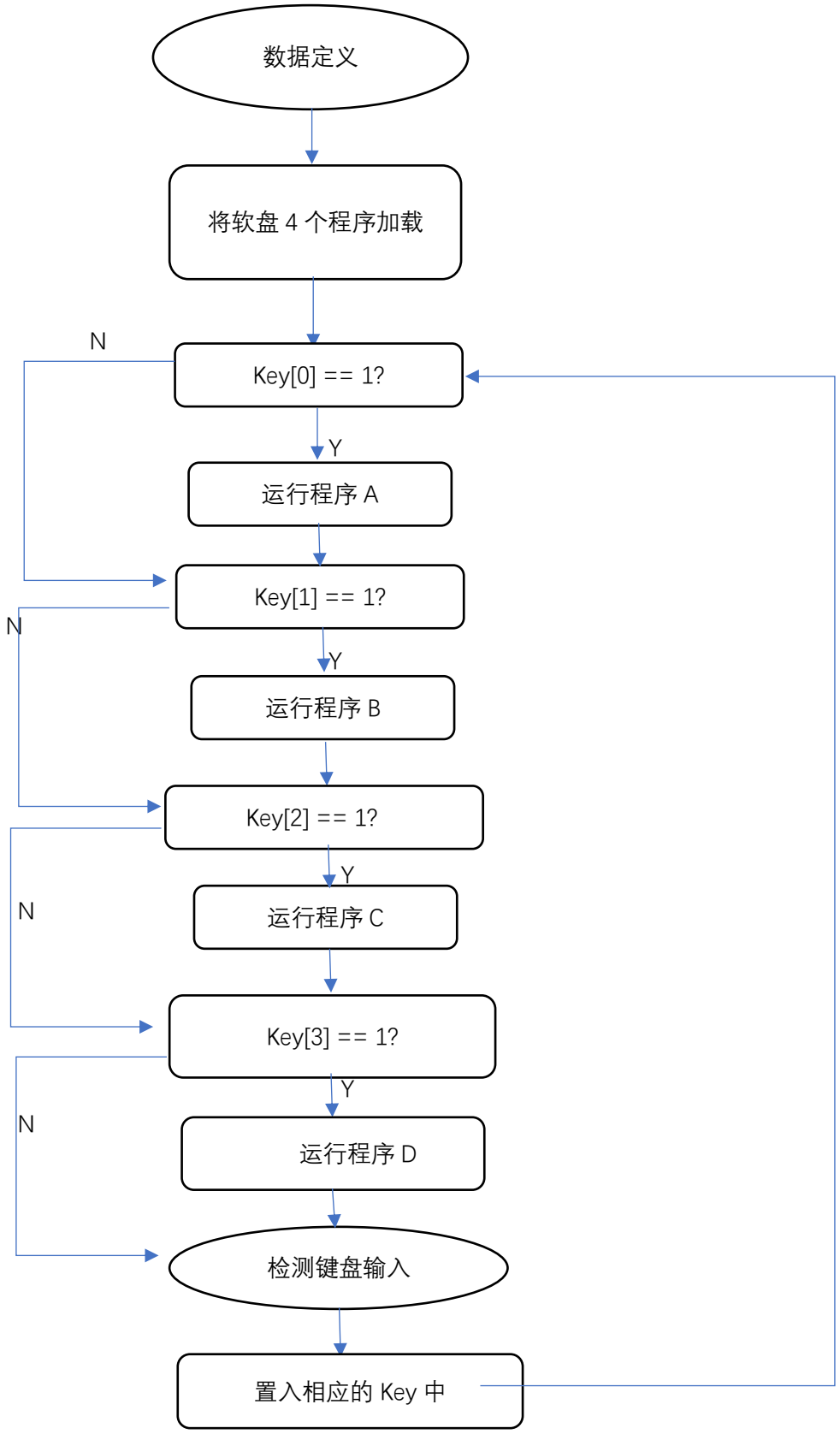
硬件：个人计算机
主机操作系统：Windows/Linux/Mac OS/其它
虚拟机软件：VMware/VirtualPC/Bochs/其它
PC 虚拟机裸机/DOS 虚拟机/其它

实验开发工具

汇编语言工具：x86 汇编语言
高级语言工具：标准 c 语言
磁盘映像文件浏览编辑工具
调试工具：Bochs

（3）程序流程

注：在本次试验中，为了让字符能成功同时动起来，我运用了简单分时处理



(4)程序模块功能

该程序分为几大模块：

1.监控程序：通过分式操作实现四个程序的运行

2.子程序 A,B,C,D（分别控制四分之一屏幕上的字符弹跳）

监控程序又分为几大模块：

1.软盘程序导入模块（INT 13H 实现）

2.分时运行模块（使得不同区域的字符同时弹跳）

3.检测输入模块（由 INT 16H 实现）

(5)代码文档组成

此次实验中，四个字程序均是由实验一中的程序演变而来，因此没有太大的变化，监控程序代码如下：

1.数据定义：

```
org  7c00h      ; BIOS将把引导扇区加载到0:7C00h处，并开始执行
OffSetA equ 8100h
OffSetB equ 8600h
OffSetC equ 9100h
OffSetD equ 9600h
```

正如实验一中所知，我们使用 org 指令将代码加载到内存的 7c00H 处作为监控程序 OffSetA, OffSetB, OffSetC, OffSetD 分别是四个子程序加载到的内存单元首地址

2.修改段寄存器并显示首行字符：

此处程序修改段寄存器后，运用 INT 10H 显示 Message 字符串

```
    mov ax, cs          ; 置其他段寄存器值与CS相同
    mov ds, ax          ; 数据段
    mov bp, Message       ; BP=当前串的偏移地址
    mov ax, ds        ; ES:BP = 串地址
    mov es, ax        ; 置ES=DS
    mov cx, MessageLength  ; CX = 串长(=9)
    mov ax, 1301h         ; AH = 13h(功能号)、AL = 01h(光标置于串尾)
    mov bx, 0007h         ; 页号为0(BH = 0) 黑底白字(BL = 07h)
    mov dh, 0               ; 行号=0
    mov dl, 0             ; 列号=0
    int 10h           ; BIOS的10h功能:显示一行字符
```

```
Message:
    db 'Press A,B,C,D to run/stop the program '
MessageLength  equ ($-Message)
```

3.加载软盘程序到内存中：

此处运用 INT 13H 中断指令来加载软盘扇区的程序，注意，在该指令入口中，c1 为起始扇区号，a1 为扇区数。因此，此时的子程序并不需要有监控程序 512 字节的限制了。

像这样的程序共有四个，因为要加载四个软盘程序到内存中。

```
LoadnA:
    ;读软盘或硬盘上的若干物理扇区到内存的ES:BX处:
     mov ax,cs                  ;段地址 ; 存放数据的内存基地址
     mov es,ax                  ;设置段地址(不能直接mov es,段地址)
     mov bx, OffSetA  ;偏移地址; 存放数据的内存偏移地址
     mov ah,2                   ; 功能号
     mov al,1                   ;扇区数
     mov dl,0                   ;驱动器号 ; 软盘为0, 硬盘和U盘为80H
     mov dh,0                   ;磁头号 ; 起始编号为0
     mov ch,0                   ;柱面号 ; 起始编号为0
     mov cl,2                   ;起始扇区号 ; 起始编号为1
     int 13H ;                  调用读磁盘BIOS的13h功能
     ; 用户程序a.com已加载到指定内存区域中
```

4.判断是否运行子程序：

```
start:
      mov bl,[Key]
      cmp bl,1
      jnz jumpA
      call OffSetA
jumpA:
      mov bl,[Key+1]
      cmp bl,1
      jnz jumpB
      call OffSetB
jumpB:
      mov bl,[Key+2]
      cmp bl,1
      jnz jumpC
      call OffSetC
jumpC:
      mov bl,[Key+3]
      cmp bl,1
      jnz jumpD
      call OffSetD
jumpD:
```

正如我在程序流程图中所画的那样，在该段程序中，我通过 Key 字段的值来判断是否应该运行该处子程序，若不运行，则跳过该子程序，并进行下一次判断

5.检测键盘输入并改变相应的 Key 值

```
jumpD:
      mov ah,1
      int 16h
      jz start    ;若没有键盘输入，则回到start阶段，重新开始
      mov bx,ax
      mov bh,0
      mov al,[Key+bx-41h]   ; 输入A,B,C,D使得Key字段中的值改变
      cmp al,1                ;若原来的值为1 ，则将其改变为0
      jnz one
      mov byte[Key+bx-41h],0   ;若原来的值为0，则将其改变为1
input:
      mov ah,0
      int 16h
      jmp start
one:
      mov byte[Key+bx-41h],1
      jmp input
```

此处程序根据键盘输入来改变相应的 Key 值；
比如：键盘输入 A，若 key[0] = 0; 则使得 key[0] = 1;反之，则变为 0
这样程序就能通过键盘输入来控制四个程序的运行与停止。

6.数据定义：

```
Message:
      db 'Press A,B,C,D to run/stop the program '
MessageLength  equ ($-Message)
Key:
      db 0,0,0,0
```

四。实验过程与实验结果

1.编译代码文件，与写入软盘二进制文件操作基本相同，此处不再累赘，仅给出部分二进制软盘代码截图：

```
00000080  01 75 03 E8 7A 09 8A 1E  EA 7C 80 FB 01 75 03 E8  .u.鑪.?陉€?u.??
00000090  6E 14 8A 1E EB 7C 80 FB  01 75 03 E8 62 19 B4 01  n.?鼸€?u.鑒.???
000000A0  CD 16 74 CA 89 C3 B7 00  8A 87 A7 7C 3C 01 75 0B  ?t菁梅.妊  <.u.?
000000B0  C6 87 A7 7C 00 B4 00 CD  16 EB B3 C6 87 A7 7C 01  茂  .??氤茂  ..?
000000C0  EB F3 50 72 65 73 73 20  41 2C 42 2C 43 2C 44 20  膝Press A,B,C,D
000000D0  74 6F 20 72 75 6E 2F 73  74 6F 70 20 74 68 65 20  to run/stop the
000000E0  70 72 6F 67 72 61 6D 20  00 00 00 00 00 00 00 00  program ........
000000F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000110  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000120  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000140  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000150  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000160  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000190  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001A0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001B0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001C0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001D0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001E0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 55 AA  ..............U?
00000200  8C C8 8E D8 B8 00 B8 8E  C0 FF 0E C0 82 75 FA C7  屑庁?笋?.纪u  U?
00000210  06 C0 82 50 C3 FF 0E C2  82 75 EE C7 06 C0 82 50  .纪P?.聧u钋.纪P?
00000220  C3 C7 06 C2 82 44 02 B0  01 3A 06 C4 82 74 1C B0  们.聧D.?:.臆t.??
00000230  02 3A 06 C4 82 74 66 B0  03 3A 06 C4 82 0F 84 AE  .:.臆tf?:.臆.劗?
00000240  00 B0 04 3A 06 C4 82 0F  84 F2 00 FF 06 C5 82 FF  .?:.臆.匀.  .解
00000250  06 C7 82 8B 1E C5 82 B8  0D 00 29 D8 74 0E 8B 1E  .蒛?解?.)豺.?解
00000260  C7 82 B8 25 00 29 D8 74  18 E9 1F 01 C7 06 C5 82  蒛?.)豺.?.?解解
00000270  0B 00 83 3E C7 82 25 74  16 C6 06 C4 82 02 E9 0A  ..?蒛%t.?臆.?解
00000280  01 C7 06 C7 82 23 00 C6  06 C4 82 04 E9 FC 00 C7  .?蒛#.?臆.辄.?锋
00000290  06 C7 82 23 00 C6 06 C4  82 03 E9 EE 00 FF 0E C5  .蒛#.?臆.轭.  .?
000002A0  82 FF 06 C7 82 8B 1E C7  82 B8 25 00 29 D8 74 0E  ?.蒛?蒛?.)豺.  .?
000002B0  8B 1E C5 82 B8 00 00 29  D8 74 18 E9 CD 00 C7 06  ?解?.)豺.鐇.? .?
```

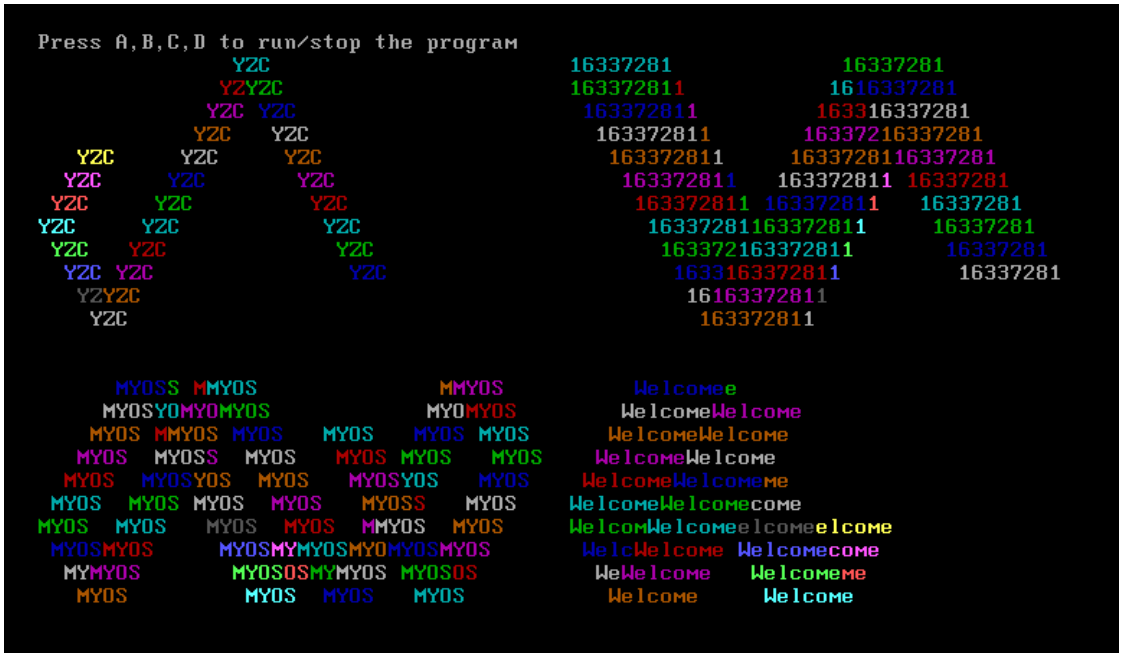
注：在本次试验中，一共有四个子程序，分别是 A,B,C,D 他们分别控制左上，右上，左下，右下的屏幕中字符的弹射。**进入操作系统后，输入任意大写的 A,B,C,D 便可启动或停止任意的程序。本次实验中，四个程序是可以同时运行的。**

2.打开虚拟机运行，显示效果:

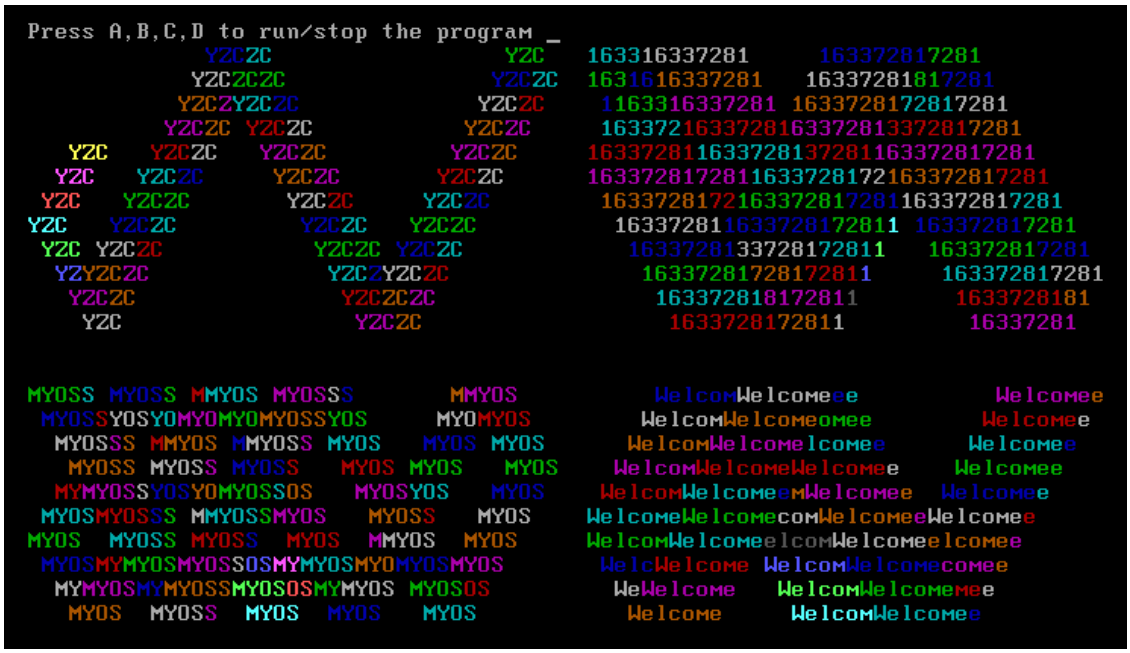
先输入大写的 A，左上角程序开始运行，然后再输入一次大写 A，程序停止：

接下来输入大写的 B，右上角程序开始运行，然后再输入一次大写 B，程序停止：

接下来连续输入 C,D 可以看到，下方两个代码一起开始运行：



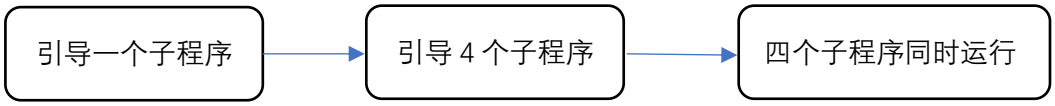
总之，想要运行哪个程序，就输入哪个字母，要停止某个程序的话，就再输入一次即可，在我设计的实验二中，四个程序甚至可以同时运行，效果如下：



至此处，说明本次实验设计成功，达到预期目的。

五、实验总结：

1.本次实验是第二次操作系统试验，相比于第一次的生疏和茫然，这一次我的实验过程显得更有条理。与第一次实验相同，我采取了循序渐进的方法，一次性引导四个子程序可能稍有难度，但是一开始只引导一个子程序就使得任务更加简单。所以我本次实验的步骤可以用以下流程图表示：



其实，根据循序渐进这个思路来讲，我觉得很符合操作系统试验课的大体思路：**从简单的程序逐渐进化为一个操作系统**。因此我以后也会将此思想贯通到我的每次实验当中。

接下来说说**本次试验遇到的一些问题：**

（1）子程序如何返回监控程序？

一开始我想的是用 jmp，但是仔细一想，这种思路显然不可成立，因为 jmp 指令无法跳到另一个程序的代码段。在思索过后我果断采用了 CALL-RET 指令 CALL 指令本身就是为了调用子程序而诞生的，而进入子程序后，只需要 RET 指令即可返回到 CALL 指令的下一指令。

CALL 指令的执行过程：

第一步：先将 call 指令的下一条指令的 CS 和 IP 入栈（当然如果是段间转移就要将 CS 和 IP 入栈，如果是段内转移就只要将 IP 入栈）

第二步：就是操作与 call 对应的 jmp 指令

RET 指令的执行过程:

通俗地讲，RET 指令就是 CALL 指令的倒序过程。

（2）如何将小球同时弹跳

我们都知道，在单核 CPU 中，要执行两个程序是比较困难的，需要保护环境，以及分时处理等等操作，但是很巧的事情是，在本次实验当中，我们子程序的运行并不需要保护环境这个操作，只需要将四个子程序进行简单分时处理即可。即为先处理一个程序接下来处理另外一个程序，周期性进行。即将处理机控制权在四个程序之间不停切换就能实现。

（3）为什么不能连续 INT 16H?

一开始使用 INT 16H 这个指令的时候，我直接不动脑的全部将 ah 置为 1，这样子会遇到一个问题：键盘输入输入一次之后，键盘缓冲区未清空。因此 ah=0 和 ah=1 的情况需要混合使用。

（4）有关代码的保存问题。

这是一个让我哭笑不得的问题，因为在本次试验中，我花了不少功夫写好了代码，本以为大功告成了。但是一天之后 360 软件自动清理垃圾时，将我的代码删除了，于是一切重头再来。以后会尽量避免这种让人啼笑皆非的事情，遇到这种事情也不是一次两次了。。。。。。。。。

2.实验感想

本周的实验是写一个监控程序，监控程序是最早期的操作系统的形式，这可以让我们从本次试验中粗略窥见监控程序的魅力。

这次试验中我对 INT 16H 中 AH 端口的疑惑很大，经过网上的检索解决了我的问题。这再一次证明了网络的力量是无穷的，我以后也应该更加善于利用网络，解决自己的困难与疑惑。

本学期开始之时，本以为操作系统是一门十分无趣的课程，但是在这次实验当中我发现，克服的困难越大，成就感也就越大。这大概和解决一个很难的数学问题一样，越难的问题越能让人有成就感。相信以后的实验难度也会慢慢变大，我认为拥有这种知难而上的精神是必要的。希望能从这门课中，了解学习到操作系统的精髓与原理，并运用到实验课程之中。