山东大学

信息科学与工程学院

2020－2021 学年第二学期

实 验 报 告

课程名称：　微处理器原理与应用

实验名称：　实验 4　实现三个程序的编写

专　业　班　级　2019 级崇新学堂

学　生　学　号　201900121023

学　生　姓　名　李禹申

实　验　时　间　2021 年 4 月 6 日

实验报告

【实验目的】

1. 在上次实验完成之后，经历了洗礼这次再练习三个实验，观察训练效果如何

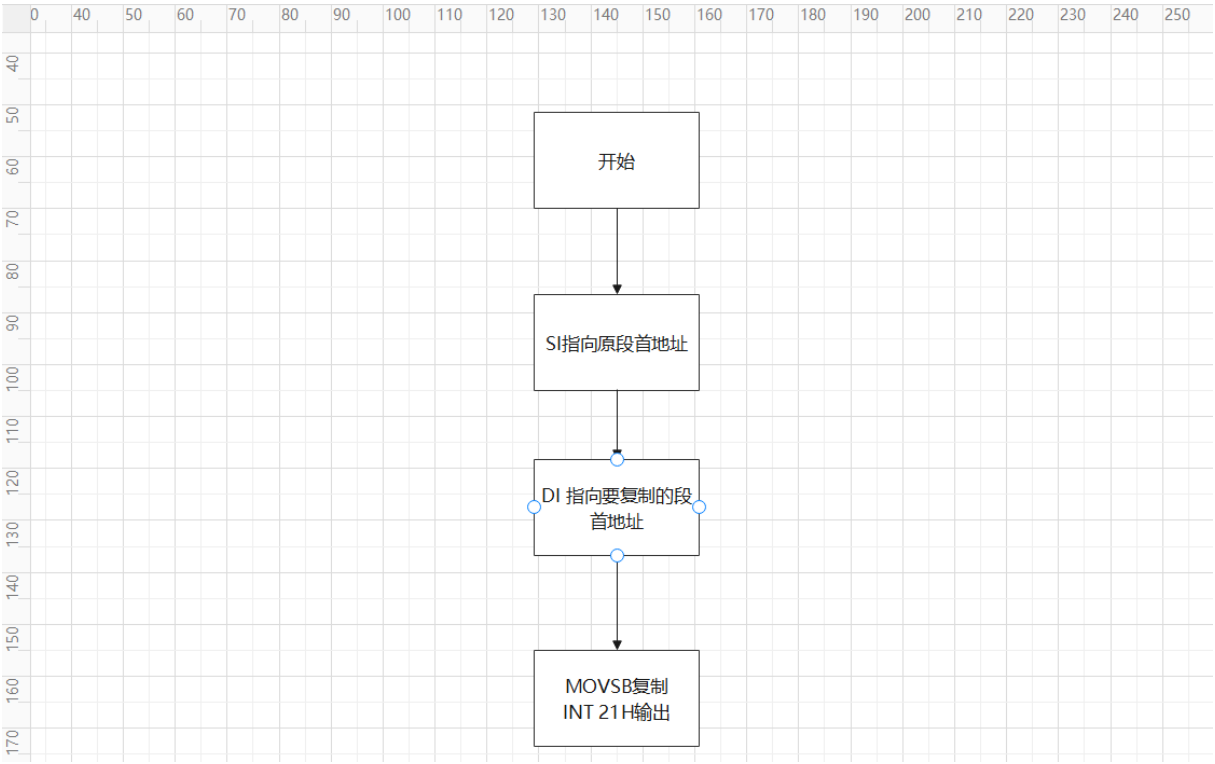
【实验要求】

1. 自己先进行编写，上网查找资料进行修改
2. 每一段代码都注释好

【实验具体内容】

【第一个实验】

（1）实验流程图（从实验 2.2 分支程序实验和循环程序实验开始必须画流程图）：



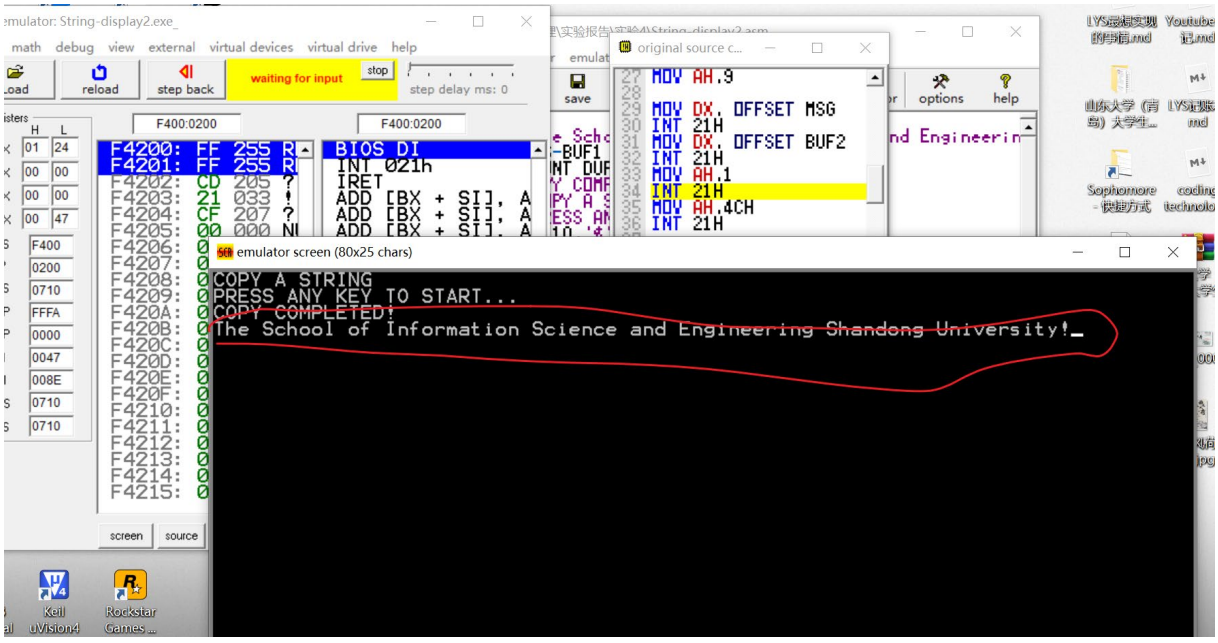
（2）实验源代码（粘贴源代码）：

```
DATAS SEGMENT
    BUF1 DB 'The School of Information Science and Engineering Shandong University!$'
    ;这个代码好啊，直接把我那个无法判断多长的问题解决了
    COUNT EQU $-BUF1
    BUF2 DB COUNT DUP(0)
    ;一些提示信息
    MSG DB 'COPY COMPLETED!$'
```

```
    MSGB DB 'COPY A STRING',13,10
         DB 'PRESS ANY KEY TO START...'
         DB 13,10,'$'
DATAS ENDS

CODES SEGMENT
    ASSUME CS:CODES,DS:DATAS,ES:DATAS
START:
    MOV AX,DATAS
    MOV DS,AX
    MOV ES,AX



    MOV AH,9
    ;这样写和用 LEA 是一样的，不过为了训练新知识就先这样写
    MOV DX, OFFSET MSGB
    INT 21H
    MOV AH,1
    INT 21H

   CALL COPY

    MOV AH,9

    MOV DX, OFFSET MSG
    INT 21H
    MOV DX, OFFSET BUF2
    INT 21H
    MOV AH,1
    INT 21H
    MOV AH,4CH
    INT 21H

    ;执行复制代码功能的子程序
```

```
COPY PROC

;根据 bloc 中提到的形式进行复制
MOV SI,OFFSET BUF1
MOV DI,OFFSET BUF2
;指定循环次数
MOV CX,COUNT
;在字符串操作中，能够使得 SI DI 增加，指向后面的地址
CLD
REP MOVSB
;从上述指令继续向下执行
RET
COPY ENDP

CODES ENDS
    END START
```

（3）实验代码、过程、相应结果（截图）并对实验进行说明和分析



成功复制

【实验心得】

1.思路：首先该程序想要实现字符串的复制功能，那么如何复制呢？想到了之前像数组一样，一个一个读字符串中的字符，然后再把字符放入。那在汇编中，我们可以读取 data 段中的字符，一个一个的读，放入栈段中，再采用 FILO 的方式输出，而倒序输

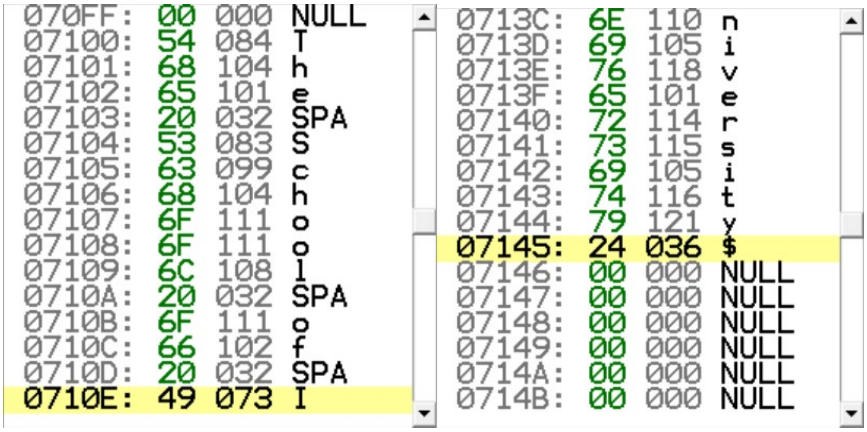
出和正序输出就按照相反的顺序 push 和 pop 一下 data 的代码段吧。OK！just do it~

2.

```
ASSUME CS:CODES,DS:DATAS,SS:STACKS,ES:EXT
```

再复习一下这里的寄存器指向，CS（CODE SEGMENT） DS（DATA SEGMENT）SS（STACK SEGMENT）ES（EXTRA SEGMENT）

3.那么承接上面的思路，如何把字符压入栈呢，当然首先得知道每个字符占多少字节，在 emu8086 中找到 ds 寄存器的起始地址，找到数据。这个时候我就在想，诶？！我应该如何使得程序能够压栈的时候自动知道字符串到头了呢，就是用字符串结束标志$来判断，读图可知，通过比较所得字符的 16 进制数，当为 24h 的时候就跳转到下一步



4.当我想要使用 cx 和 bx 作为计数器的时候，发现改变 cx 的值是会使得程序无法正常进行的，我运行了一下注销掉的程序，发现在程序运行过程中 cx 的值是不断地改变的，而 dx 和 bx 中的值始终为 0000，所以姑且改用 dx 和 bx

```
    ;call duplicate
    ;call turn_duplicate

duplicate proc near
    lea si,string_a - 1
    mov sp,0000h
    ;mov cx,0
tt1:inc si
    ;inc cx
    mov ax,[si]
    push ax
    cmp ax,24h
    je output
    loop tt1

output:
    lea si,string_b - 1
    ;mov bx,0
tt2:inc si
    ;inc bx
    pop [si]
    mov ax,[si]
    cmp bx,dx
    je  display
    loop tt2
```

5.编程的时候有点太累了，我就想着上网借鉴一下大神们的代码吧，结果发现

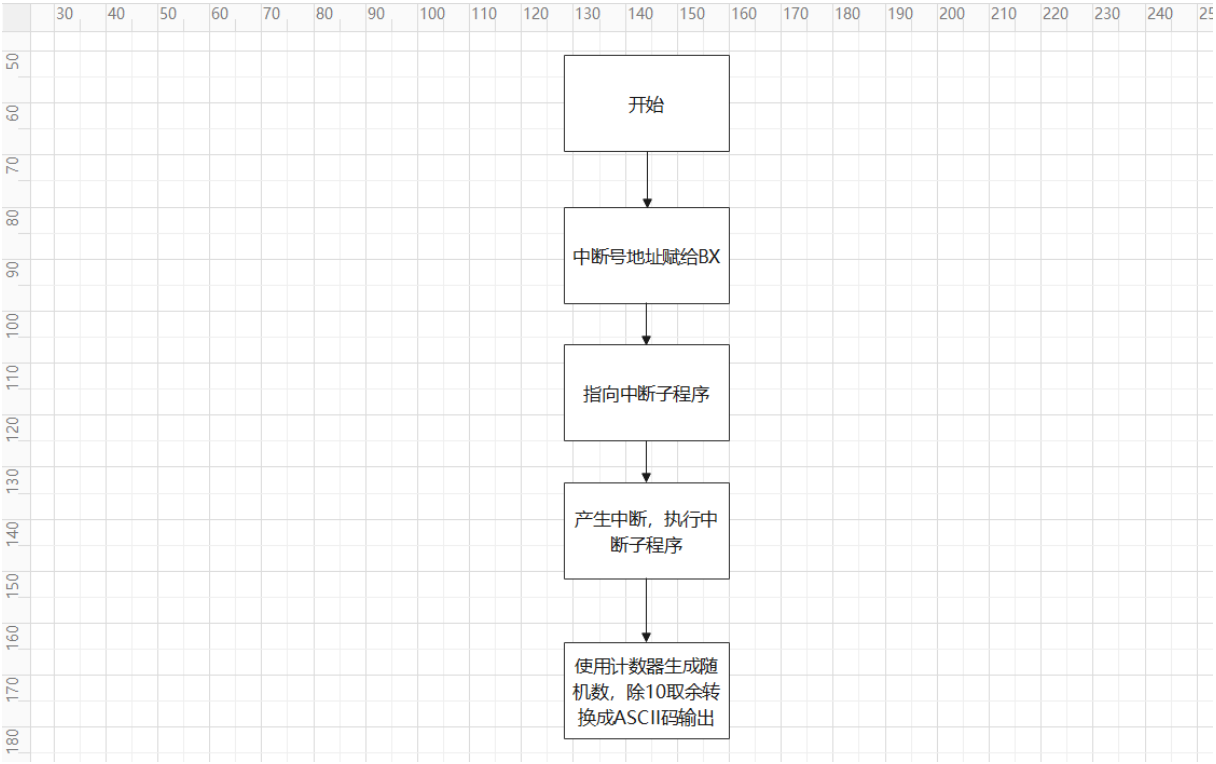
> 今天写了一个小程序,实现了把字符串从存储器的一个单元拷贝到存储器的另一个单元,程序中主要用到了字符串操作指令MOVSB,这个指令比较特殊可以对两个存储单元直接操作,用SI指向源字符串地址,DI指向符加段的目地字符串地址,用这条指令就可以很轻松的将字符串进行拷贝,如果你想循环操作,很简单,在指令前面加上REP伪指令便可实现指定次数的循环,但是要求你把你想要循环的次数送到CX,这样就可以复制字符串了.示例代码如下:

原来还有这么简单的指令，我赶紧学了起来，仔细研读了一下代码，改成了自己需要的样子

【实验心得】上学期在学 python 的时候，做那个 EECS 我就非常的痛苦，就不知道怎么提升自己的代码水平，然后就发现实际上代码不仅仅是写出来就够了，更为重要的是改到完美，而怎么改到完美呢，就是借鉴大神的代码……当然，我这个实验属于完全被 CSDN 那个大神给征服了，因为实在是比我的优美太多了，但是这个如何进行倒叙输出，我再想想

【第二个实验】

（1）实验流程图：



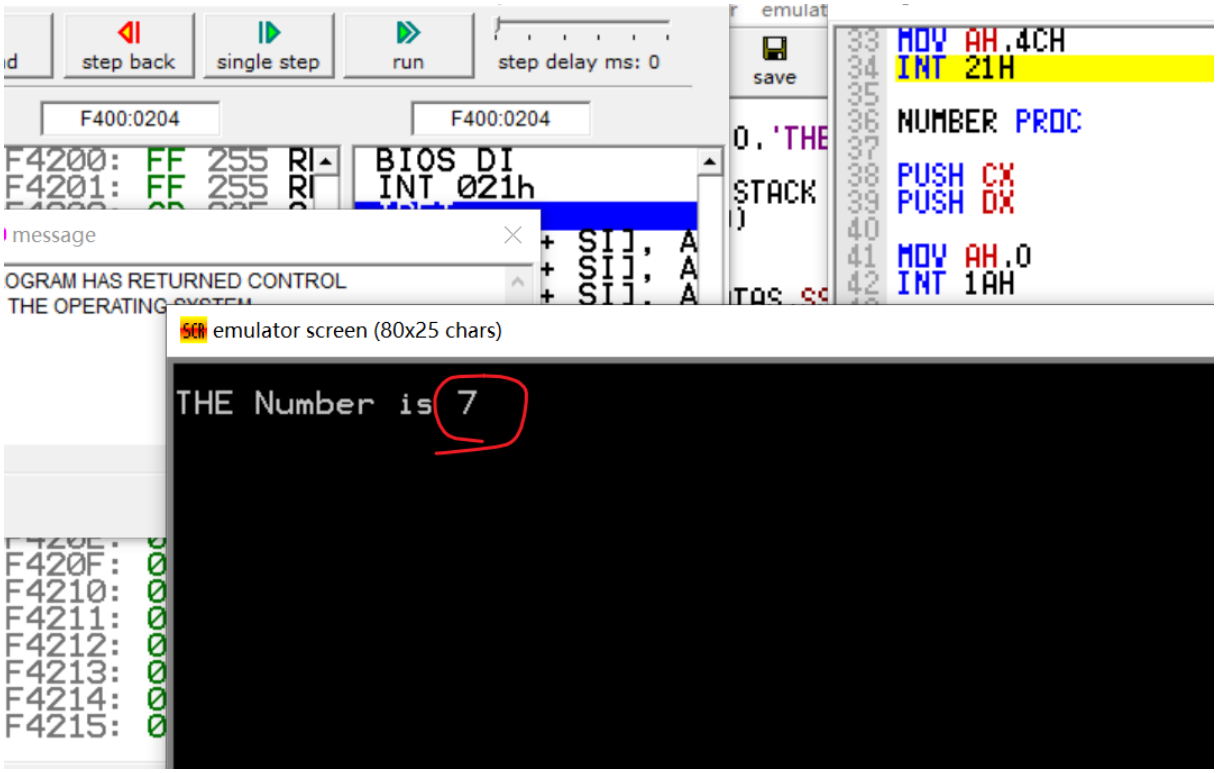
（2）实验源代码（粘贴源代码）：

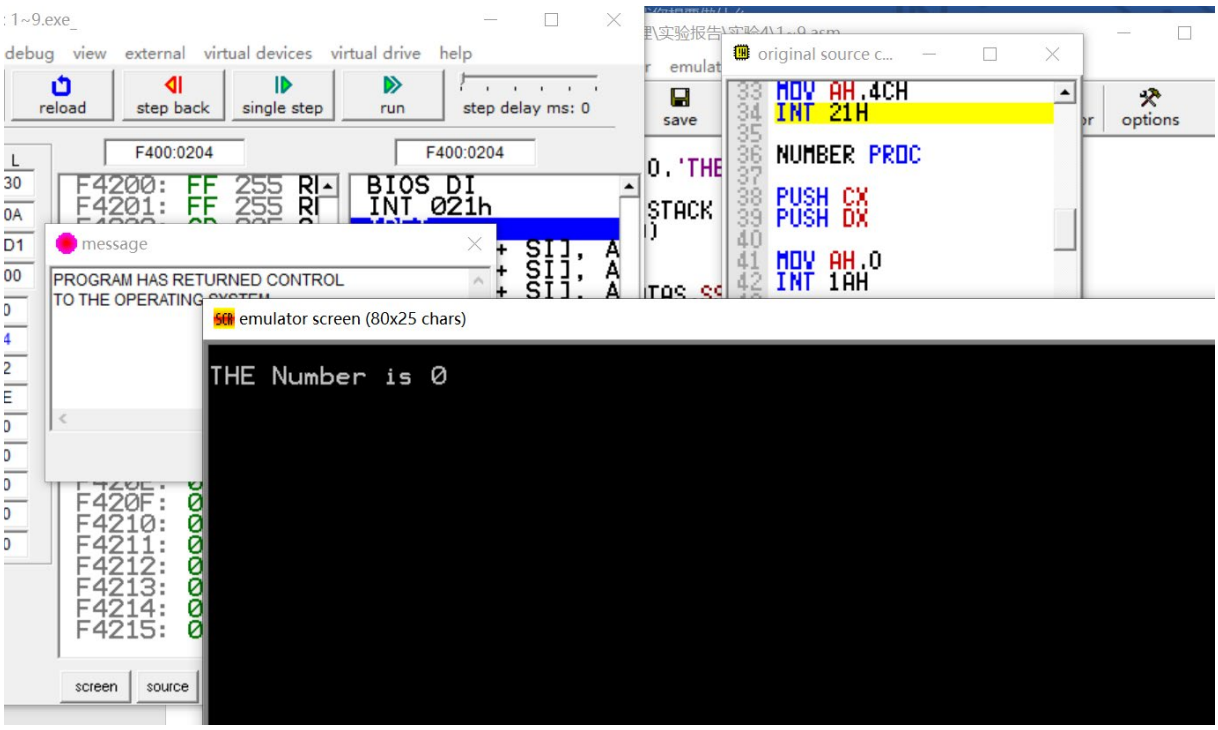
```
DATAS SEGMENT
    BUF1 DB 13,10,'THE Number is ','$'
DATAS ENDS
STACKS SEGMENT STACK
    DB 100 DUP(0)
STACKS ENDS
CODES SEGMENT
    ASSUME DS:DATAS,SS:STACKS,CS:CODES

START:
    MOV AX,DATAS
    MOV DS,AX

    ;显示提示字符串
    LEA DX BUF1
    MOV AH,9
    INT 21H
```

```
    MOV AX,0
    MOV DS,AX
    ;把中断号 86H 的地址赋给 BX
    MOV BX,86H*4
    ;禁止中断发生
    CLI
    ;进行字操作.将 MAKERAND 的偏移地址存入 ES:BX
    MOV WORD PTR DS:[BX],OFFSET NUMBER
    ;进行字操作.将 MAEKRAND 的段基地址存入 ES:[BX+2]
    MOV WORD PTR DS:[BX+2],SEG NUMBER
    ;允许中断发生
    STI
    ;调用 86H
    INT 86H    ;调用 86,NUMBER 子程序

    MOV AH,4CH
    INT 21H

NUMBER PROC
    ;备份
    PUSH CX
    PUSH DX

    MOV AH,0
    INT 1AH     ;读时钟计数器值.CX:DX 存着计数器值

    ;一位数只保留低 16 位，DX 赋值成 0
    MOV AX,DX
    MOV DX,0
    MOV BX,10
    DIV BX
    ;转换成 ASCII 码
    ADD DL,30H
```

```
    MOV AH,02H
    INT 21H

    ;恢复数据
    POP DX
    POP CX
    IRET
NUMBER ENDP

CODES ENDS
END START
```

（3）实验代码、过程、相应结果（截图）并对实验进行说明和分析：





成功生成了随机数

【实验心得】

1.首先该程序的重点在于中断子程序的书写，但是前提是能够通过中断进入到这个中断子程序中，所以我上网找到了一个中断子程序的固定书写格式，当时我在实验 3 就找了，如下是我按照格式书写的进入中断子程序的代码

```
MOV AX,0
MOV DS,AX
;把中断号86H的地址赋给BX
MOV BX,86H*4
;禁止中断发生
CLI
;进行字操作.将MAKERAND的偏移地址存入ES:BX
MOV WORD PTR DS:[BX],OFFSET MAKERAND
;进行字操作.将MAEKRAND的段基地址存入ES:[BX+2]
MOV WORD PTR DS:[BX+2],SEG MAKERAND
;允许中断发生
STI
```

2.接下来就是针对中断子程序的书写了，想要实现随机数的输出实际上非常简单，在网上查到到了一种生产随机数的方法是由内部的计数器产生随机数

利用计算机内部计数器生成真随机数原理及实现

原创 置顶 sjd163 2010-02-22 06:56:00 1290 收藏 2

文章标签： byte 任务 汇编 语言 delete 工作

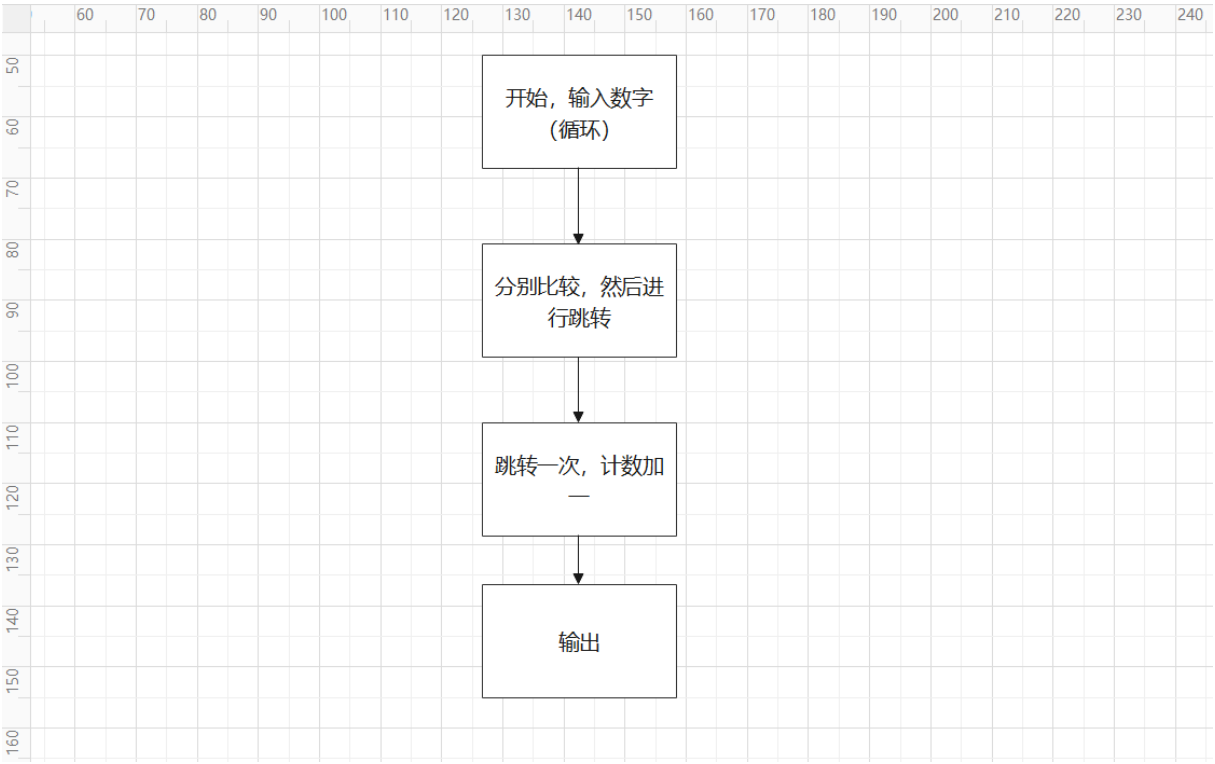
但是我对其中相应的原理并不是很了解，所以就在那里各种组合试一下，加上百度，发现除 10 取余是最简单的办法。

【实验心得】

中断子程序的调用复习了上课所讲的内容，1~9 随机数用计数器来生成十分的新鲜有意思，很好玩

【第三个实验】

（1）实验流程图：



（2）实验源代码（粘贴源代码）：

```
DATAS SEGMENT
    BUF1 DB 10,13,'PLEASE INPUT SCORES:','$'
    SCORE6 DB 10,13,'SCORE6:','$'
    SCORE7 DB 10,13,'SCORE7:','$'
    SCORE8 DB 10,13,'SCORE8:','$'
    SCORE9 DB 10,13,'SCORE9:','$'
    SCORE10 DB 10,13,'SCORE10:','$'
```

```
    SCOREOUT DB 10,13,'OSOCRE:','$'
    SSCORE6 DB 30H
    SSCORE7 DB 30H
    SSCORE8 DB 30H
    SSCORE9 DB 30H
    SSCORE10 DB 30H
    OUTSCORE DB 30H
DATAS ENDS
STACKS SEGMENT
    DB 300 DUP(0)
STACKS ENDS
CODES SEGMENT
    ASSUME CS:CODES,DS:DATAS,SS:STACKS
START:
    MOV AX,DATAS
    MOV DS,AX
    MOV AH,9

    LEA DX,BUF1
    INT 21H

    MOV CX,10
INPUT: MOV AH,1
    INT 21H
    ;调用程序进行比较，因为仅凭十位就可以确定区间了
    CALL COMPARE
    ;输入第二个数，就和分析的一样实际上是没用的，为了好看吧……反正
    INT 21H
    ;显示空格
    MOV AH,2
    MOV DL,32
    INT 21H
    ;循环输入每一个成绩
    LOOP INPUT ;循环输入每一个成绩
```

```
    CALL OUTPUT;调用子程序进行输出
    MOV AH,4CH
    INT 21H       ;程序结束
;比较最高位数值的子程序
COMPARE PROC
    PUSH AX
    ;如果最高位等于 1（这里实际上忽略了百分位）则跳转到 s10
    CMP AL,31H
    JZ S10
    ;以下都同上
    CMP AL,36H
    JZ S6
    CMP AL,37H
    JZ S7
    CMP AL,38H
    JZ S8
    CMP AL,39H
    JZ S9
    JMP S0

;计数
S0:
    ADD [OUTSCORE],1
    JMP exit
S6:
    ADD [SSCORE6],1
    JMP exit
S7:
    ADD [SSCORE7],1
    JMP exit
S8:
    ADD [SSCORE8],1
    JMP exit
S9:
```

```
    ADD [SSCORE9],1
    JMP exit
S10:
    ADD [SSCORE10],1
    MOV AH,1
    INT 21H

exit:POP AX
    RET
COMPARE ENDP

;定义子程序 OUTPUT
OUTPUT PROC
    ;针对 6 开头的输出
    LEA DX,SCORE6        ;输出提示
    MOV AH,9
    INT 21H

    MOV DL,[SSCORE6] ;输出个数
    MOV AH,2

    INT 21H                          ;回车
    MOV DL,10
    INT 21H

    ;以下均同上
    LEA DX,SCORE7
    MOV AH,9
    INT 21H
    MOV DL,[SSCORE7]
    MOV AH,2
    INT 21H
    MOV DL,10
    INT 21H
```

```
LEA DX,SCORE8
MOV AH,9
INT 21H
MOV DL,[SSCORE8]
MOV AH,2
INT 21H
MOV DL,10
INT 21H

LEA DX,SCORE9
MOV AH,9
INT 21H
MOV DL,[SSCORE9]
MOV AH,2
INT 21H
MOV DL,10
INT 21H

LEA DX,SCORE10
MOV AH,9
INT 21H
MOV DL,[SSCORE10]
MOV AH,2
INT 21H
MOV DL,10
INT 21H

LEA DX,SCOREOUT
MOV AH,9
INT 21H
MOV DL,[OUTSCORE]
MOV AH,2
INT 21H
```

RET

OUTPUT ENDP

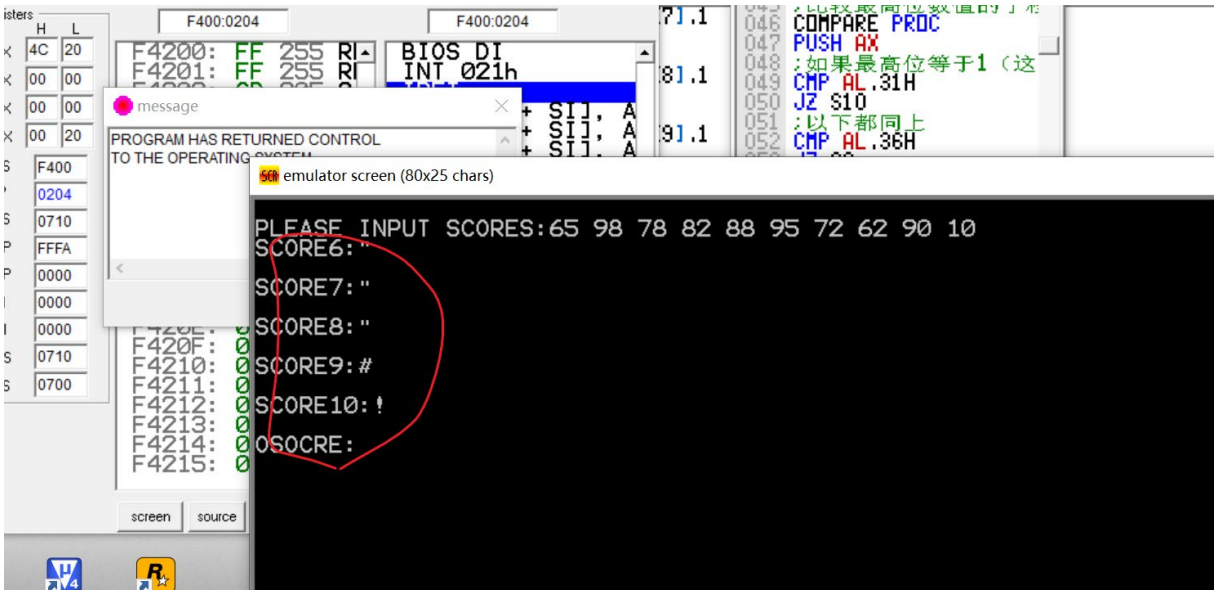
CODES ENDS

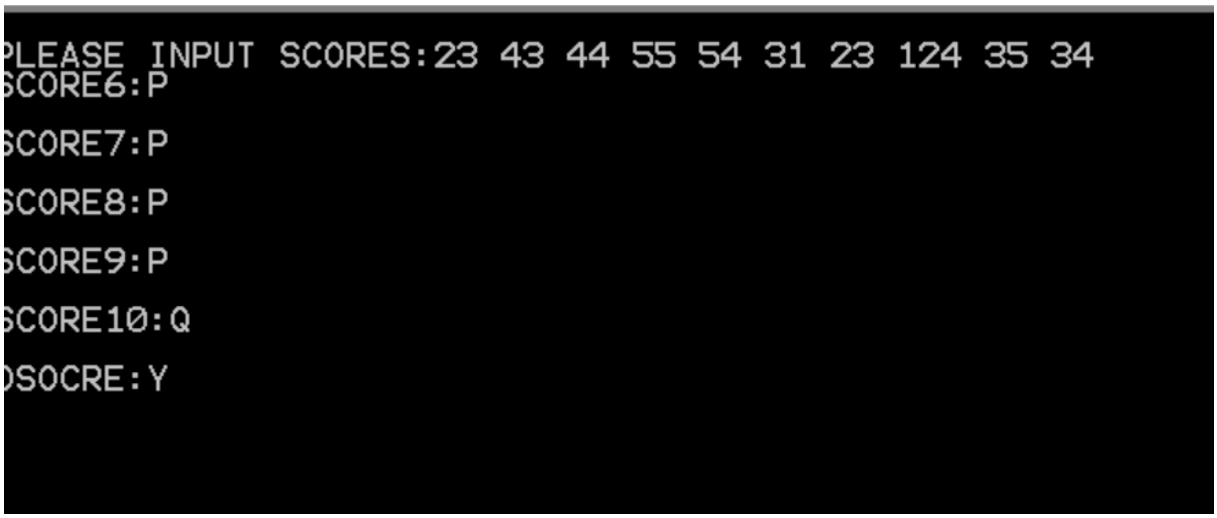
END START

（3）实验代码、过程、相应结果（截图）并对实验进行说明和分析：

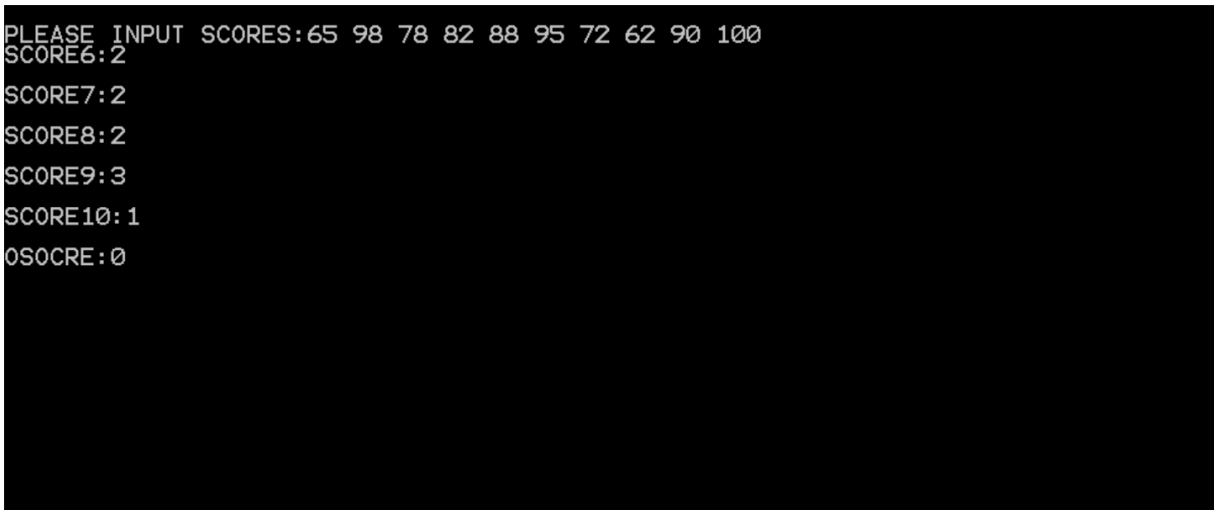
【实验过程】

第一次编写之后出错了，心态崩了

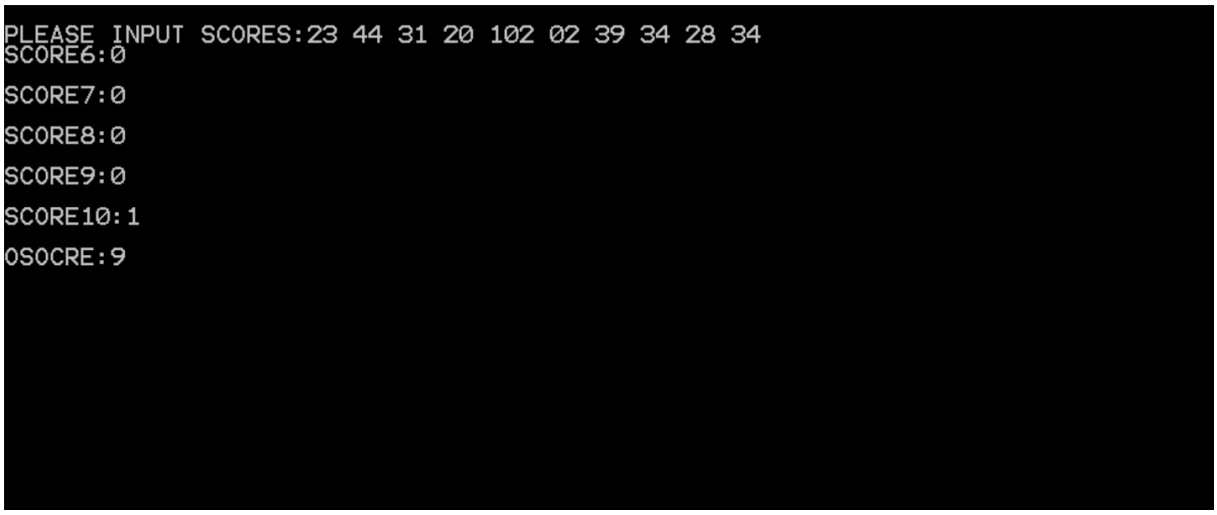




哦哦哦！我知道了，我可以在之前设置 30H，然后进行相加直接输出 ASCII 码（网上学的）



随便输入一个



【实验分析】

首先，当我看到这个题目的时候我的第一想法是，无论怎么说都是分两个步骤而已，第一步：分类并统计，第二步：放入。第一步的分类实际上非常简单，只需要看十位数的大小就可以了，这也就是之前字符输出的那里面判断 1~9 字符的种种情况而已，然后跳转到相应的子程序即可。放入就更加地简单，在 DATAS 数据中定义好各个 SCORE 然后放入内存中就 ok 了！bingo 开始行动，EZ

【实验心得】

这个将 ASCII 转换进行提前统一放置，非常的开心，真是一个好办法，直接就可以输出 ASCII 码

【总体实验心得】

汇编实验结束了，我感觉给我最大的感悟就是，在写汇编之前总是想骂他，想砸电

脑，写完之后又很有成就感，就好像一个孩子长大了一般。我学到最重要的东西就是“程序是改出来的”我学会了先将程序的基本功能实现，再去网上找大神的代码，进行对比和学习，虽然很多时候都是我去改成类似他们的样子，但是那个过程中我真的学到了很多，不断的反思自己。汇编实验做的很开心，谢谢老师，完结撒花