

目录

1 题目一	4
1.1 题目描述	4
1.2 主要算法说明	4
1.2.1 磁盘调用	4
1.2.2 读取文件	6
1.2.3 统计字符	8
1.2.4 打印结果	11
1.3 运行界面的其他功能	13
1.4 运行结果	17
1.4.1 起始界面	17
1.4.2 登录成功界面	18
1.4.3 登录失败界面	18
1.4.4 统计询问界面 (Y)	19
1.4.5 统计询问界面 (N)	19
1.4.6 文件输入与统计输出	20
1.4.7 文件名错误	21
1.5 课程设计总结	22
1.6 参考文献	23
 2 题目二	 24
2.1 题目描述	24
2.2 主要算法说明	24
2.2.1 宏操作定义	24
2.2.2 n 值的输入	29
2.2.3 阶乘的计算	30
2.2.4 阶乘结果的输出	33
2.3 运行界面	34
2.3.1 起始界面	34

2.3.2	登录成功	34
2.3.3	n 值输入并计算	35
2.3.4	退出程序	35
2.4	课程设计总结	36
2.5	参考文献	36
3	题目三	37
3.1	题目描述	37
3.2	实验原理	37
3.3	实验步骤	39
3.3.1	选择实验器材	39
3.3.2	搭建实验电路	39
3.3.3	初始化各芯片的控制信号	40
3.3.4	写存储器	40
3.3.5	读存储器	41
3.4	运行界面	42
3.5	课程设计总结	42
3.6	参考文献	42
4	题目四	43
4.1	题目描述	43
4.2	实验原理	43
4.3	实验设计	45
4.4	运行界面	46
4.5	课程设计总结	46
4.6	参考文献	46
5	整体总结	46
6	附录	48

6.1	题目一源代码	48
6.2	题目二源代码	61

1 题目一

1.1 题目描述

题目：给定一个英文 ASCII 码文件，统计文件中英文字母的频率，以十进制形式输出

基本要求：对于给定英文 ASCII 码文件，统计文件中每个英文字母的次数，计算的每个英文字母频率，以十进制形式输出每个英文字母对应的频率

1.2 主要算法说明

1.2.1 磁盘调用

在本实验中，需要能够读取文件。在汇编语言中，可以通过 int 21h 中断调用来实现。在本实验中，需要调用的是 int 21h 中断的 3dh 功能，即读取文件。其调用格式如下：

```
1 mov ah, 3dh
   mov al, 0
3 mov dx, offset filename
   int 21h
```

中断调用

以上只是一般的文件调用，在本实验中，打开文件后还需要对文件进行各种操作，如读取文件、关闭文件等。因此从最开始的输入文件名开始分析。

输入文件名 输入文件名需要在 data 区设置一个文件名字符串，用来接收用户输入的文件名，输入代码如下：

```
2 ;输入文件名
   lea dx,filename
   mov ah,0ah
```

```
4      int 21h
```

输入文件名

对文件名进行操作并打开文件 在这里对文件名进行操作，获取文件名的长度，然后在文件名后添加 0，调用 int 21h 中断的 3dh 功能，打开文件。代码如下：

```
2      mov bl,filename+1;获取文件名长度
      mov bh,0
      mov [bx+filename+2],0;文件名后加0
4      lea dx,filename+2
      mov ax,3d00h;打开文件
6      int 21h
```

对文件名进行操作并打开文件

异常处理 在文件打开过程中，可能会出现文件名输入错误造成无法打开的情况，因此需要对这种情况进行处理。在这里，如果文件打开失败，会返回一个错误代码，可以通过调用子过程实现。代码如下：

```
      jnc open
2      mov si,offset error1;打开文件失败
      call dmess
4      jmp continue
```

异常处理

其中，dmess 是一个子过程，用来输出错误信息，代码如下：

```
dmess proc
2 dmess1:
      mov dl,[si]
4      inc si
      or dl,dl
6      jz dmess2
```

	mov ah,02h
8	int 21h
	jmp dmess1
10	dmess2:
	ret
12	dmess endp

dmess

通过打印错误信息，可以让用户知道错误的原因，然后子过程结束，跳转到 continue 标签处，重新输入文件名。

1.2.2 读取文件

打开文件后，对文件进行读取，在这里调用 readchar 子过程进行单一字符的读取，然后调用 punch 子过程对字符出现频率进行统计，同时对读取异常进行处理；如果读取到文件尾部，则跳转 typeok 并打印读取到的字符串，代码如下：

	go:
2	call readchar; 读取一个字符
	jc readerror; 读取失败
4	cmp al, eof; 判断是否到文件尾
	jz typeok; 到文件尾，显示结果
6	call punch; 统计字符
	jmp go; 继续读取

读取文件

readchar 子过程 readchar 子过程用来读取一个字符，调用 int 21h 中断的 3fh 功能，代码如下：

1	readchar proc
	mov cx, 1
3	mov dx, offset buffer

	mov ah,3fh
5	int 21h
	jc r1
7	cmp ax,cx
	mov al,eof
9	jb r2
	mov al,buffer
11	
	r2:
13	clc
	r1:
15	ret
	readchar endp

readchar 子过程

其中，将读取到的字符存储在 buffer 中，如果读取失败，则返回一个进位标志，如果读取成功，则返回一个进位标志和一个零标志，如果读取到文件尾，则返回一个进位标志和一个负标志。

punch 子过程对字符串进行计数，同时在技术之前会打印该字符，在后面将会单独展开。

异常处理 读取字符时遭遇读取异常，调用 dmess 子过程返回错误信息，代码如下：

	readerror:
2	mov si,offset error2
	call dmess

异常处理

读取完成 当读取到 eof 时，跳转到 typeok 标签处，关闭文件，然后打印统计到的结果，代码如下：

1	typeok:
---	---------

```

mov ah,3eh;关闭文件
3   int 21h
mov dl,0ah
5   mov ah,2
int 21h
7   call show

```

读取完成

1.2.3 统计字符

统计字符是本实验中最核心的功能，需要对每个字符出现的次数进行统计以便后面的打印。在这里，使用了一个 26 个元素的数组来存储每个字符出现的次数，data 段定义如下：

```

1   array db 26 dup(0)

```

定义数组

在统计字符前，先打印该字符，保留 dx 中原有值，需要将 dx 中的元素进栈，然后再打印该字符，代码如下：

```

1   push dx
mov dl,al
3   mov ah,02h
int 21h
5   pop dx

```

打印字符

在统计字符时，本实验采取的是不区分大小写，因此在判断的时候需要对大小写分别进行判断，采用 ASCII 码进行字符的判断，需要对字符的 ASCII 码区间进行划分，划分如下：

1. 小写字母：97-122
2. 大写字母：65-90

3. 其他

上述 ASCII 码值使用的是 10 进制，在进行比较的时候采用 16 进制。

第一轮比较 在上述读取过程中，将读取到的字符存在了 al 中，将 al 中的元素存在 ch 中，然后将 41h 存在 cl 中，先进行一次比较，如果小于则跳转其他字符计数，实现代码如下：

```
1      mov cl,41h
      lea di,array
3      mov ch,al
      cmp ch,cl
5      jb other
```

第一轮比较

然后将 ch 中的值与 5ah 进行比较，如果大于则跳转 higher2，准备进行第二轮比较，实现代码如下：

```
1      cmp ch,5ah
      ja higher2
```

第一轮比较

在确定了是大写字母后，进行字符的定位，如果 cl、ch 不匹配，则将 cl++，再进行匹配，直到匹配成功，跳转 char 处进行计数，实现代码如下：

```
      h1:
2      je char
      ja loop1
4
loop1:
6      inc cl
      add di,1
8      jmp h1
```

第一轮比较

第二轮比较 第二轮比较与第一轮比较类似，只是将 41h 改为 61h，5ah 改为 7ah，实现小写字母的统计，实现代码如下：

```
higher2:
2   mov cl,61h
   lea di,array
4   mov ch,al
   cmp ch,cl
6   jb other
   cmp ch,7ah
8   ja other

10  h2:
   cmp ch,cl
12  je char
   ja loop2
14

16  loop2:
   inc cl
   add di,1
18  jmp h2
```

第二轮比较

其他字符 如果不是大写字母和小写字母，则跳转到 other 标签处，实现代码如下：

```
other:
2   inc others
```

其他字符

字符计数 无论是大写字母还是小写字母，都会跳转到 char 标签处进行计数，采用记录偏移量的方式，将 array 的偏移量记录在 di 中，并将 di 中内容赋给 ch 来实现 ++，实现代码如下：

```

char:
2  sub ch,ch
   mov ch,[di]
4  inc ch
   mov [di],ch

```

字符计数

1.2.4 打印结果

在打印结果时，需要将结果转换为十进制，然后再打印。打印时按照顺序，并默认是大写输出，将 array 中的元素依次取出，并存在 al 中，通过调用 display 子过程进行十进制转化然后再打印，实现代码如下：

```

1  show proc
   lea si,array
3  mov di,41h
   loop3:
5  lea dx,string1
   mov ah,09h
7  int 21h
   mov dx,di
9  mov ah,02h
   int 21h
11 lea dx,string2
   mov ah,09h
13 int 21h
   sub ax,ax
15 mov al,[si]
   add si,1
17 call display
   call endlne
19 inc di
   cmp di,5bh

```

```

21         jb loop3
           ret
23 show endp

```

打印结果

display 子过程 display 子过程用来将十进制转换为字符串，然后再打印，实现代码如下：

```

1 display proc near
   mov bl,10
3   div bl
   push ax
5   mov dl,a1
   add dl,30h
7   mov ah,02h
   int 21h
9   pop ax
   mov dl,ah
11  add dl,30h
   mov ah,02h
13  int 21h
   mov dl,20h
15  mov ah,02h
   int 21h
17  ret
display endp

```

display 子过程

endlime 子过程 endlime 用来调整打印的格式，实现代码如下：

```

2 endlime proc near
   mov dl,20h
   mov ah,02h

```

```

4      int 21h
      mov dl,20h
6      mov ah,02h
      int 21h
8      mov dl,20h
      mov ah,02h
10     int 21h
      mov dl,20h
12     mov ah,02h
      int 21h
14     mov dl,20h
      mov ah,02h
16     int 21h
      ret
18 endl ine endp

```

endl ine 子过程

1.3 运行界面的其他功能

在本实验中，还加载了其他功能，如登录、统计询问、退出程序等功能，下面将对这些功能进行介绍。

登录 在本实验中额外添加了登录过程，实现原理比较简单，在 data 段定义了一个用户名和密码，然后在程序开始时，进入登录界面，输入用户名和密码，如果正确则跳转到主界面，如果错误则跳转到登录失败界面，代码如下：

```

      input:
2      lea dx,username
      mov ah,09h
4      int 21h

6      lea dx,tempname

```

```

8      mov ah,0ah
      int 21h

10     cmp byte ptr tempname+1,05h
      jnz repeat1

12     mov cx,5

14     mov si,offset user
      mov di,offset tempname+2

16     mov ax,dataarea
      mov es,ax

18     cld
      repe cmpsb

20     jnz repeat1

22     mov dx,offset tempname+2 ;显示输入的字符串
      mov byte ptr tempname[7], '$'

24     call dosshow

26     lea dx,password
      mov ah,09h

28     int 21h

30     lea dx,temppassword
      mov ah,0ah

32     int 21h

34     cmp byte ptr temppassword+1,06h
      jnz repeat2

36     mov cx,6

38     mov si,offset pass
      mov di,offset temppassword+2

40     mov ax,dataarea
      mov es,ax

```

```

42      cld
      repe cmpsb
44      jnz repeat2

46      mov dx,offset temppassword+2
      mov byte ptr temppassword[8], '$'
48      call dosshow

50      jmp loginsuccess

```

登录

统计询问 在开始统计前，先进行询问，如果不需要统计则结束程序，如果需要统计则跳转到统计过程，代码如下：

```

      request:
2      lea dx,starting;显示提示信息
      mov ah,09h
4      int 21h

      mov ah,01h;获取键盘输入
      int 21h

8      cmp al,'N';判断是否统计字符
10     je finish;不统计字符，结束程序
      cmp al,'n'
12     je finish

      cmp al,'Y';统计字符
14     je continue
      cmp al,'y'
16     je continue

18
      mov dl,0ah;回车换行
20     mov ah,2

```

	int 21h
22	mov dl,0dh
	mov ah,2
24	int 21h
26	jmp request; 输入错误, 重新输入

统计询问

其中, continue 标签处是统计过程, finish 标签处是结束程序。

continue 标签 在 continue 标签处, 就提示输入文件名, 然后调用打开文件过程, 代码如下:

	continue:
2	mov dl,0ah
	mov ah,2
4	int 21h
	mov dl,0dh
6	mov ah,2
	int 21h
8	
	lea dx,string; 显示提示信息
10	mov ah,09h
	int 21h
12	...

continue 标签

finish 标签 在 finish 标签处, 显示结束信息, 然后调用 4ch 中断, 结束程序, 代码如下:

	finish:
2	lea dx,ending; 显示结束信息
	mov ah,09h
4	int 21h

6	<pre>mov ah,4ch;结束程序 int 21h</pre>
---	------------------------------------

finish 标签

1.4 运行结果

1.4.1 起始界面

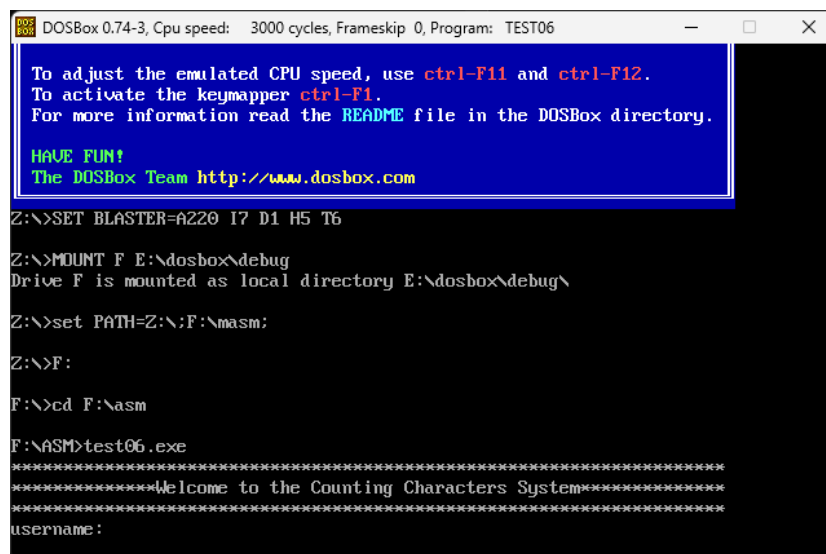
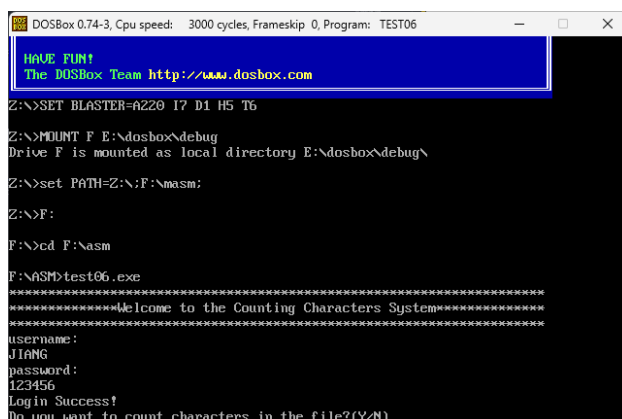


图 1: 起始界面

1.4.2 登录成功界面



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TEST06
HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=a220 I7 D1 H5 T6

Z:\>MOUNT F E:\dosbox\debug
Drive F is mounted as local directory E:\dosbox\debug\

Z:\>set PATH=Z:\F:\nasm:

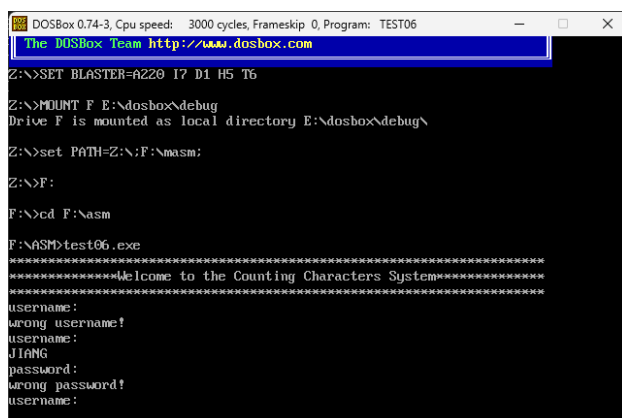
Z:\>F:

F:\>cd F:\nasm

F:\ASM>test06.exe
*****Welcome to the Counting Characters System*****
username:
JIANG
password:
123456
Log in Success!
Do you want to count characters in the file?(Y/N)_
```

图 2: 登录成功界面

1.4.3 登录失败界面



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TEST06
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=a220 I7 D1 H5 T6

Z:\>MOUNT F E:\dosbox\debug
Drive F is mounted as local directory E:\dosbox\debug\

Z:\>set PATH=Z:\F:\nasm:

Z:\>F:

F:\>cd F:\nasm

F:\ASM>test06.exe
*****Welcome to the Counting Characters System*****
username:
wrong username!
username:
JIANG
password:
wrong password!
username:
```

图 3: 登录失败界面

1.4.4 统计询问界面 (Y)

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TEST06
HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=a220 I7 D1 H5 T6

Z:\>MOUNT F E:\dosbox\debug
Drive F is mounted as local directory E:\dosbox\debug\

Z:\>set PATH=Z:\F:\nasm:

Z:\>F:

F:\>cd F:\nasm

F:\ASM>test06.exe
*****Welcome to the Counting Characters System*****
username:
JIANG
password:
123456
Login Success!
Do you want to count characters in the file?(Y/N)Y
Input the file name:_
```

图 4: 统计询问界面 (Y)

1.4.5 统计询问界面 (N)

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TEST06
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=a220 I7 D1 H5 T6

Z:\>MOUNT F E:\dosbox\debug
Drive F is mounted as local directory E:\dosbox\debug\

Z:\>set PATH=Z:\F:\nasm:

Z:\>F:

F:\>cd F:\nasm

F:\ASM>test06.exe
*****Welcome to the Counting Characters System*****
username:
JIANG
password:
123456
Login Success!
Do you want to count characters in the file?(Y/N)N
Thank you for using the counting characters system
```

图 5: 统计询问界面 (N)

1.4.6 文件输入与统计输出

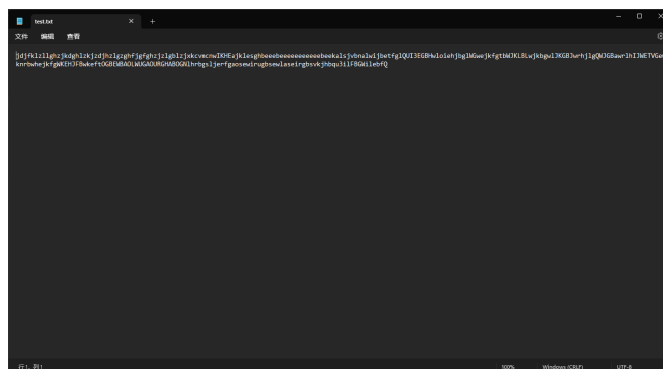


图 6: 待测文本

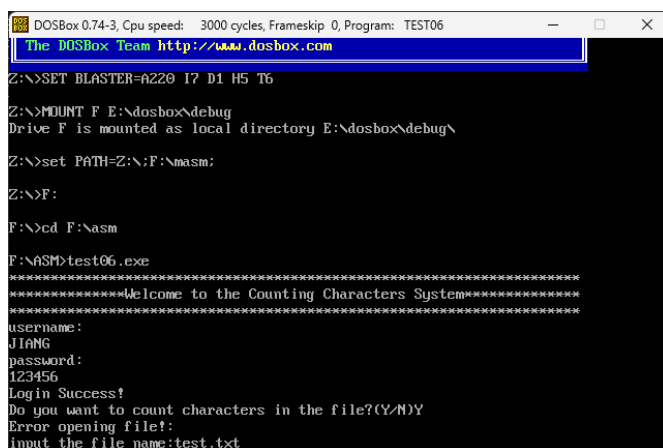


图 7: 文件输入

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TEST06
F:\ASM>test06.exe
*****Welcome to the Counting Characters System*****
username:
JIANG
password:
123456
Login Success!
Do you want to count characters in the file?(Y/N)Y
Error opening file!:
jd jfk1z1lghz jkdghlzk jzd jhzlgzghf jgfgghz jzlgblz jkxcvncwIKHEa jklesghbeeebeeeeee
eebeeka ls jobna lwi jbetfg iQU13EGBHwloieh jbg lWgwe jkfgtbWJLKLBLw jkbqwlJKGBJwrh jlgQWJG
Bawr lhIJWETUGewlknrbuhe jkfgWKEHJFBwkef tOGBEWBAOLWUGAOURGHABOGNlhrbgs ljerfgaosewi
rughsewaseirghsvk jhbqu3l lFBGWi lebfQ
number of A:06      number of B:15      number of C:02      number of D:03
number of E:28      number of F:09      number of G:18      number of H:12
number of I:06      number of J:18      number of K:12      number of L:22
number of M:01      number of N:03      number of O:02      number of P:00
number of Q:01      number of R:07      number of S:07      number of T:03
number of U:02      number of V:03      number of W:13      number of X:01
number of Y:00      number of Z:09
Counting success!
Do you want to count characters in the file?(Y/N)_
```

图 8: 统计输出

1.4.7 文件名错误

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TEST06
The DOSBox Team http://www.dosbox.com
Z:\>SET BLASTER=A220 I7 D1 H5 T6
Z:\>MOUNT F E:\dosbox\debug
Drive F is mounted as local directory E:\dosbox\debug\
Z:\>set PATH=Z:\;F:\asm:
Z:\>F:
F:\>cd F:\asm
F:\ASM>test06.exe
*****Welcome to the Counting Characters System*****
username:
JIANG
password:
123456
Login Success!
Do you want to count characters in the file?(Y/N)Y
Error opening file!:111
input the file name: _
```

图 9: 文件名错误

1.5 课程设计总结

问题与解决问题的过程

1. 对文件的操作不熟悉。在使用 masm2015 集成环境的时候，无论将文件放置在哪个安装目录下都无法打开文件，程序始终返回打开错误信息，然后选择使用 dosbox 手动编译、链接、打开程序，成功实现了对文件的操作
2. 界面不够美观。在实验过程中，由于 dosbox 的年代比较久远，UI 界面比较难优化，在调对齐的时候反复编译，最终使统计结果打印的整齐

收获与感悟 统计字符串的实验给我带来了以下收获：

- 熟悉汇编语言：通过实验，我能够更加熟悉汇编语言的语法和基本指令。这对于理解计算机底层的工作原理非常重要，可以加深对计算机体系结构的理解。
- 理解字符编码：在统计英文字母频率的实验中，我需要对 ASCII 码进行处理。这让我更加深入地理解了字符编码的概念，以及不同编码方案对应的字符集。
- 掌握文件处理技巧：实验要求读取文件中的内容并统计英文字母的频率，这让我学会了在汇编语言中处理文件的基本技巧，包括打开文件、读取文件内容和关闭文件等。
- 提高问题解决能力：在实验过程中，我可能会遇到一些问题，例如文件读取失败、数据处理错误等。通过解决这些问题，我可以提高自己的问题解决能力和调试技巧。
- 培养耐心和细致：汇编语言编程需要非常细致和耐心，因为每个指令都需要仔细编写，并确保没有错误。通过实验，我可以培养这种耐心和细致的工作态度。

总的来说，通过完成这个汇编实验，我可以进一步加深对计算机底层的理解，学会处理文件和字符编码，并培养解决问题的能力 and 耐心。这些都对我未来的学习和职业发展有着积极的影响。

1.6 参考文献

《汇编语言程序设计》雷向东，雷振阳，龙军中南大学出版社
实验源码较长，因此将附录统一置于报告最后！

2 题目二

2.1 题目描述

题目：用递归计算 $n!$ ($n \geq 50$)，以十进制数输出基本要求：输入一个不小于 50 的整数 n ，用递归计算 $n!$ ，以十进制数输出

2.2 主要算法说明

在子程序嵌套的情况下，如果一个子程序调用的子程序就是它自身，这样的子程序称为递归子程序。求解 $N!$ 本身是一个子程序，又因为 $N! = N * (N-1)!$ ，所以求解 $N!$ 的子程序可以调用求解 $(N-1)!$ 的子程序，这样的子程序就是递归子程序。递归子程序的特点是：在子程序中调用自身，且每次调用时所用的参数不同。

递归通常使用堆栈来保存临时参数。递归调用展开时，保存在堆栈中的数据就有用：假设给定任意 n ，即可计算 $n-1$ 的阶乘。这样就可以不断减少 1，直到它等于 0 为止。根据定义， $0! = 1$ 。而回溯到原始表达式 $n!$ 的过程，就会累积每次的乘积，直到 $n=0$ 为止。

2.2.1 宏操作定义

为了简化代码，同时减少提高代码的复用性，本实验中定义了一些宏操作。

OUTPUTCHAR 为了能够输出字符，定义了一个宏操作，代码如下：

```
2      ; 字符输出
      OUTPUTCHAR MACRO AINCHAR ; 将字符AINCHAR输出
4          PUSH AX
          PUSH BX
          PUSH CX
6          PUSH DX
```


8	MOV DL,AINCHAR	
	MOV AH,02H	;输出字符
10	INT 21H	
12	POP DX	
	POP CX	
14	POP BX	
	POP AX	
16	ENDM	

OUTPUTCHAR

其中，对 AX、BX、CX、DX 进行了压栈操作，以便在输出字符后能够恢复原来的值，防止数据丢失

OUTPUTSTR OUTPUTSTR 能够将一整个字符串输出，代码如下：

	;字符串输出	
2	OUTPUTSTR MACRO AIMSTR	;将字符串AIMSTR输出
	PUSH AX	
4	PUSH BX	
	PUSH CX	
6	PUSH DX	
8	LEA DX,AIMSTR	;将AIMSTR的偏移地址送到DX寄存器
	MOV AH,09H	;09H字符串输出功能
10	INT 21H	
12	POP DX	
	POP CX	
14	POP BX	
	POP AX	
16	ENDM	

OUTPUTSTR

其中，对 AX、BX、CX、DX 进行了压栈操作，以便在输出字符串后能够恢复原来的值，防止数据丢失。

OUTPUTAX OUTPUTAX 能够以十进制输出 AX 中的数值，代码如下：

```

;以10进制输出AX中的数值
2 OUTPUTAX MACRO          ;将AX中的数值以10进制形式输出
    PUSH AX
4    PUSH BX
    PUSH CX
6    PUSH DX
    CALL OUTPUTAXP        ;调用进制输出过程
8    POP DX
    POP CX
10   POP BX
    POP AX
12 ENDM

```

OUTPUTAX

其中，通过调用 COUTPUTAXP 子过程实现十进制输出，代码如下：

```

    OUTPUTAXP PROC
2    MOV DX,0
    MOV CX,0              ;用CX储存余数个数后续LOOP需要使用
4    CMP AX,0             ;判断AX中的值是否为0
    JNE OUTPUTAXF1
6    OUTPUTCHAR '0'
    JMP OUTPUTAXPEXIT
8
OUTPUTAXF1:
10   CMP AX,0             ;判断AX中的值是否为0
    JE OUTPUTAXF2         ;是则说明AX已经按位除完了
12   MOV BX,10            ;10进制
    DIV BX                ;除10
14   PUSH DX              ;将余数入栈保存
    MOV DX,0

```

```

16  INC CX                ;计数循环取得的余数个数
    JMP OUTPUTAXF1

18
20  OUTPUTAXF2:          ;循环输出取得的余数
    POP AX
    ADD AL,30H
    OUTPUTCHAR AL
    LOOP OUTPUTAXF2
24  OUTPUTAXPEXIT: RET
    OUTPUTAXP ENDP

```

COUTPUTAXP

OUTPUTNUM OUTPUTNUM 能够将数字字符串 AIMNUM 表示的数值输出，代码如下：

```

1      ;输出字符串AIMNUM所表示的数值
    OUTPUTNUM MACRO AIMNUM

3          PUSH AX
          PUSH BX
5          PUSH CX
          PUSH DX
7          PUSH SI

          LEA BX,AIMNUM;用BX存储字符串AIMNUM在DS中的首地址
          CALL OUTPUTNUMP ;调用字符串AIMNUM数值输出过程

11         POP SI
13         POP DX
          POP CX
15         POP BX
          POP AX
17     ENDM

```

OUTPUTNUM

其中，通过调用 OUTPUTNUMP 子过程实现字符串数值输出，代码如下：

```
1      OUTPUTNUMP PROC
OUTPUTNUMF1:
3      MOV SI,-2
OUTPUTNUMEND:      ;使SI指向ANS的数值结尾处
5      ADD SI,2
      MOV AX,[BX+SI]      ;测试AX是否为-1
7      CMP AX,-1
      JNE OUTPUTNUMEND      ;直到搜索到最后结尾-1
9
      SUB SI,2
11     CMP SI,-2
      JE  OUTPUTNUMEXIT      ;若为-2则说明ANS中不存在数据
13     MOV AX,[BX+SI]      ;取出ANS中的第一个数值到AX中 从低到高
      OUTPUTAX      ;将AX中的数以10进制形式输出 是最高位不需要填
          0
15
OUTPUTNUMNEXT:
17     SUB SI,2
      CMP SI,-2
19     JE  OUTPUTNUMEXIT
      MOV AX,[BX+SI]      ;取出ANS中的数值到AX中 开始判断有多少0需
          要填充
21     CMP AX,1000
      JAE OUTPUTNUMF2      ;AX中的数值大于等于1000时跳转
23     OUTPUTCHAR '0'      ;AX小于1000时先输出一个字符'0'
      CMP AX,100
25     JAE OUTPUTNUMF2
      OUTPUTCHAR '0'      ;AX小于100时再输出一个字符'0'
27     CMP AX,10
      JAE OUTPUTNUMF2
29     OUTPUTCHAR '0'      ;AX小于10时再输出一个字符'0'
```

```

31 OUTPUTNUMF2:
    OUTPUTAX          ;将AX中的数以10进制形式输出
33    JMP OUTPUTNUMNEXT ;跳转进行下一位数值的输出
OUTPUTNUMEXIT:
35    RET
OUTPUTNUMP ENDP

```

OUTPUTNUMP

2.2.2 n 值的输入

在本实验中，采用 int 21h 中断的 1 号功能来实现 n 值的输入，字符默认输入到 AL 中，对输入进行判断，如果键入回车，则输入结束，如果键入数字，则将键入的数字 ascii 码减去 30h，然后乘以 10，再加上原来的值，实现代码如下：

```

    TYPEIN: ;输入需要求解的n值
2    PUSH AX
    MOV AH,01H
4    INT 21H          ;字符默认输入到AL中
    CMP AL,13
6    JE TYPEINEXIT    ;检测到回车后跳转AX的输出
    SUB AL,48          ;将字符转化为对应的数值
8    MOV BH,0
    MOV BL,AL
10   POP AX
    CMP AX,0           ;当AX中的数值为0时，跳过乘法操作
12   JE TYPEINADD
    MOV CX,10
14   MUL CX           ;乘以10
TYPEINADD:
16   ADD AX,BX
    JMP TYPEIN
18   TYPEINEXIT:
    POP AX            ;将计算得到的数值出栈到AX中

```

20	POP DX	
	POP CX	
22	POP BX	
	MOV CX,AX	;求阶乘的数转至CX中

n 值的输入

其中,将输入的字符转换为数字,每次输入一位数字,已知“0”的 ASCII 码为 30h,因此将输入的字符减去 30h,即可得到对应的数字。

2.2.3 阶乘的计算

阶乘的计算是本实验中最重要的一部分,在前期处理中,已经得到了需要计算的 n 值并存储在 AX 中,接下来就是阶乘的计算。

阶乘结果寄存器的定义 题目要求阶乘的数是不小于 50 的,根据 8086 的参数可知,单一寄存器都是 16 位的,因此单一寄存器最大存储的数就是 65535,根据简单的计算得到: $8! = 40320$, $9! = 362880$,因此不可能使用单一寄存器或者多寄存器来存储阶乘的结果,因此需要定义单独的结果存储变量来存放阶乘结果,定义如下:

1	ANS DW 1,3000 DUP(-1)	;储存运算结果 存入一个1应对输入0的情况
	ANSH DW 3000 DUP(0)	;相对高位
3	ANSL DW 3000 DUP(0)	;相对低位

阶乘结果寄存器定义

其中,ANS 是结果存储变量,每一个单元能存放四位数字,ANSH 和 ANSL 在程序中的作用如下:将 ANS 中的数值与 10000 进行比较,如果小于 10000,则直接存储在 ANS 的单元中;如果大于 10000,则将该数除以 10000,商存于 ANSH 中,余数存于 ANSL 中。

1	MOV BX,1	;BX逐步求阶的乘数
	SAVENEXT:	

```

3  CMP CX,0
   JE  OUTPUTANS      ;当CX中的值为0时，输出ANS中的数值
5  PUSH CX
   MOV SI,0           ;SI指向ANS的起始位置
7
MULANS:                ;对ANS中的所有数值进行乘BX操作，乘积大于等于
   10000的部分存储到ANSH中，小于10000的部分存储到ANSL中
9  MOV AX,ANS[SI]      ;取出ANS中的数值到AX中
   CMP AX,-1
11 JE  TRANSFORM       ;直到取得的数值为-1时，跳转
   MUL BX              ;进行乘法操作
13
   PUSH CX
15 MOV CX,10000
   DIV CX              ;除法操作 除以10000
17 POP CX
19 MOV ANSL[SI],DX      ;将余数存储到ANSL中
   ADD SI,2
21 MOV ANSH[SI],AX      ;将商存储到ANSH中
23 JMP MULANS

```

数乘

然后对 ANSL、ANSH 的格式进行调整，将商调整到 ANS 的高位中，并将余数放回 ANS 的原位

```

1  TRANSFORM:          ;对ANS乘以BX得到的数值字符串ANSL和ANSH
   , 进行格式调整，并将调整后的结果存储到ANS中去
   PUSH BX             ;BX中的乘数入栈保存
3  MOV BX,0
   MOV SI,2
5
TRANSFORMF1:
7  MOV AX,ANS[BX]      ;取出ANS中的数值到AX中

```

```

9      CMP AX,-1
      JE  TRANSFORMF2      ;当ANS中的数值取完时，跳转

11     MOV AX,ANSH[BX]      ;取商到AX中
      ADD AX,ANSL[BX]      ;加上此时所在位置对应的余数

13
      CMP AX,10000          ;判断AX中的数值是否大于10000
15     JB  SAVEINTOANS      ;小于10000时直接将数值存储到ANS中
      MOV DX,0              ;大于10000时，将大于等于10000的部分存到高位
                              ;的进位中去，小于10000的部分存储到ANS中
17     PUSH CX
      MOV CX,10000
19     DIV CX
      POP CX
21     MOV ANS[BX],DX        ;小于10000的余数部分存储到ANS中
      ADD ANSH[SI],AX        ;大于10000的高位部分添加到高位的进位中去
23
      ADD BX,2              ;指针后移指向下一个数值
25     ADD SI,2
      JMP TRANSFORMF1

27
SAVEINTOANS:
29     MOV ANS[BX],AX        ;将数值存储到ANS中
      ADD BX,2              ;指针后移指向下一个数值
31     ADD SI,2
      JMP TRANSFORMF1

33
TRANSFORMF2:
35     MOV AX,ANSH[BX]      ;取出上一个商到AX中
      CMP AX,0
37     JE  TRANSFORMF3      ;若AX中的数值为0时 跳过下一步
      MOV ANS[BX],AX        ;将上一位的商添加到ANS中
39 TRANSFORMF3:
      POP BX                ;BX中的数值出栈
41     INC BX

```



```
POP CX
LOOP SAVENEXT
```

格式调整

其中，对于不同的情况进行不同的调整，如果乘积小于 10000，则直接存储到 ANS 中；如果乘积大于 10000，则将乘积除以 10000，商存于 ANSH 中，余数存于 ANSL 中，然后将 ANSH 中的数值添加到 ANS 的高位中，将 ANSL 中的数值存于 ANS 的原位中。

2.2.4 阶乘结果的输出

阶乘的结果存储在 ANS 中，调用 OUTPUTNUM 宏操作，将 ANS 中的数值输出，实现代码如下：

1

```
OUTPUTANS:          ;输出数值字符串所表示的数值
OUTPUTNUM ANS
```

阶乘结果的输出

2.3 运行界面

2.3.1 起始界面

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TEST201

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>MOUNT F E:\dosbox\debug
Drive F is mounted as local directory E:\dosbox\debug\

Z:\>set PATH=Z:\;F:\nasm;

Z:\>F:

F:\>cd F:\nasm

F:\>F:\ASM>test201.exe

*****Welcome to the Calculating N Factorial Program*****
username:
```

图 10: 起始界面

2.3.2 登录成功

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TEST201

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>MOUNT F E:\dosbox\debug
Drive F is mounted as local directory E:\dosbox\debug\

Z:\>set PATH=Z:\;F:\nasm;

Z:\>F:

F:\>cd F:\nasm

F:\>F:\ASM>test201.exe

*****Welcome to the Calculating N Factorial Program*****
username:
JIANG
password:
123456
Login Success!
Do you want to Calculate N Factorial?(Y/N):
```

图 11: 登录成功

2.4 课程设计总结

问题与解决问题的过程

1. 递归计算阶乘不熟悉：在开始编写代码时，对递归计算阶乘不熟悉，因此在编写代码时，出现了很多错误，例如：没有考虑到阶乘的结果超过 16 位的情况，没有考虑到阶乘的结果为 0 的情况等，通过不断的调试，最终解决了这些问题。
2. 大数据的存储方式的改变：在开始编写代码时，我将阶乘的结果存储在一个单一的变量中，但是后来发现，单一的变量无法存储阶乘的结果，因此需要改变存储方式，将阶乘的结果存储在一个数组中，通过数组的方式来存储阶乘的结果。

收获与感悟 通过这次的汇编课设，我发现了自己在知识掌握和实践方面存在许多不足。首先，我的知识掌握不够扎实；其次，我在实践方面缺乏经验，这导致了很多错误的发生。在开始搭建大致框架后，我逐渐进行修改，最终成功地实现了目标。在实际运行过程中，遇到了多个问题和报错，但我能够及时解决这些问题。当遇到不懂的情况时，我会上网查阅相关知识。通过不断地经历错误和解决问题，我的能力得到了很大的提升，对知识的掌握也得到了巩固和强化，同时培养了分析和解决问题的能力以及独立思考的能力。因此，这次的汇编课设给我带来了许多收获。在汇编过程中，我使用了递归调用子程序的算法，这次经历使我对这些算法思想有了更清晰的理解。通过反复练习、总结和学习命令操作，我对汇编语言有了更深入的学习。

2.5 参考文献

《汇编语言程序设计》雷向东，雷振阳，龙军中南大学出版社
实验源码较长，因此将附录统一置于报告最后！

3 题目三

3.1 题目描述

题目：存储器系统设计

基本要求：设计一个存储器系统，包括存储器的地址空间、存储单元的位数、存储单元的个数、存储器的容量、存储器的读写功能、存储器的读写速度、存储器的结构、存储器的接口等。

3.2 实验原理

在微机系统中，常用的静态 RAM 有 6116、6264、62256 等。在本实验中使用的是 6116。6116 为 2K*8 位的静态 RAM，其逻辑如图所示：

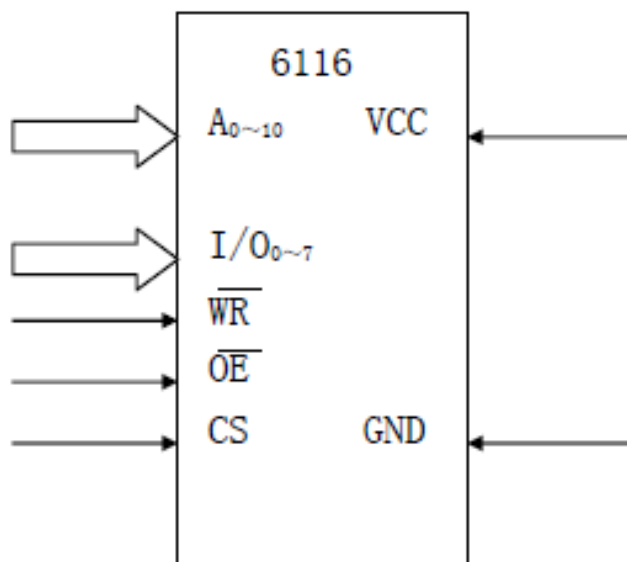


图 14: 6116 逻辑图

其中 A0~10 为 11 根地址线，I/O 为 0-7 共 8 根数据线，CS 为片选端，OE 为数据输出选通端，WR 为写信号端。6116 的工作方式如下表：

表 1: 6116 工作方式

控制信号	CS	OE	WR	数据线
读	0	0	1	输入
写	0	X	0	输出
非选	1	X	X	高阻态

实验所用的半导体静态存储器电路原理如图所示，实验中的静态存储器一片 6116（2K×8）构成，其数据线接至数据总线，地址线由地址锁存器（74LS273）给出。地址灯 AD0—AD7 与地址线相连，显示地址线内容。数据开关经一三态门（74LS245）连至数据总线，分时给出地址和数据。

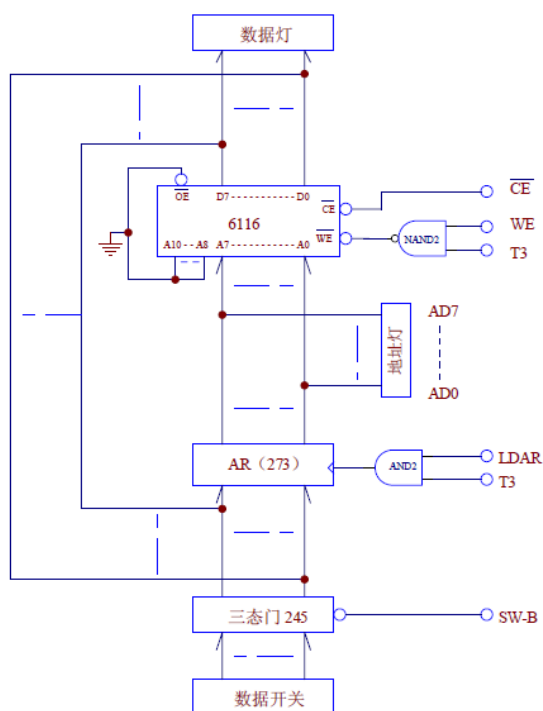


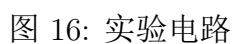
图 15: 半导体静态存储器电路原理图

因地址寄存器为 8 位，接入 6116 的地址 A7—A0，而高三位 A8—A10 接地，所以其实际容量为 256 字节。6116 有三个控制线：CE（片选线）、

3.3 实验步骤

根据实验原理图，将所需要的组件从组件列表中拖到实验设计流程栏中。

将已选择的组件进行连线（鼠标从一个引脚的端点拖动到另一组件的引脚端，即完成连线）。搭建好的实验如图所示：



3.3.3 初始化各芯片的控制信号

初始化各芯片的控制信号，仔细检查无误后点击【电源开/关】按钮接通电源。

3.3.4 写存储器

给存储器的 00、01、02、03、04 地址单元中分别写入数据 11H、12 H、13 H、14 H、15 H。

由存储器实验原理图看出，由于数据和地址全由一个数据开关给出，因此要分时地给出。下面的写存储器要分两个步骤，第一步写地址，先关掉存储器的片选（CE=1），打开地址锁存器门控信号（LDAR=1），打开数据开关三态门（SW-B=0），由开关给出要写入的存储单元的地址，双击单脉冲产生 T3 脉冲将地址输入到地址锁存器；第二步写数据，关掉地址锁存器门控信号（LDAR=0），打开存储器片选，使之处于写状态（CE=0，WE=1），由开关给出此单元要写入的数据，，双击单脉冲产生 T3 脉冲将数据写入到当前的地址单元中。写其他单元依次循环上述步骤。

写存储器流程如图所示（以向 00 号单元写入 11H 为例）。

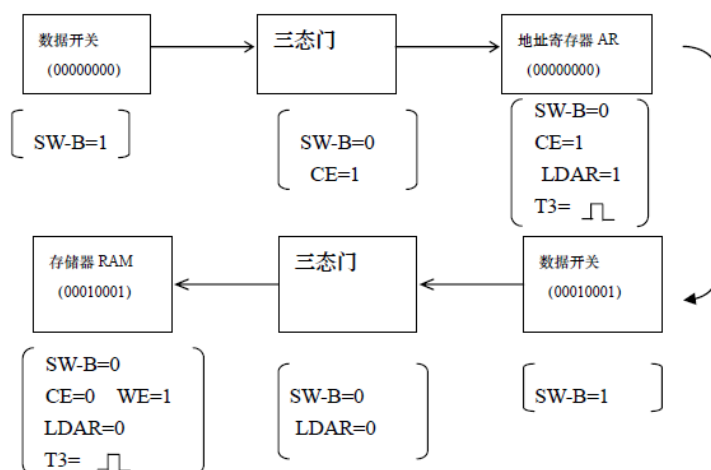


图 17: 写存储器流程

3.3.5 读存储器

依次读出第 00、01、02、03、04 号单元中的内容，观察上述各单元中的内容是否与前面写入的一致。同写操作类似，读每个单元也需要两步，第一步写地址，先关掉存储器的片选（CE=1），打开地址锁存器门控信号（LDAR=1），打开数据开关三态门（SW-B=0），由开关给出要写存储单元的地址，双击单脉冲产生 T3 脉冲将地址输入到地址锁存器；第二步读存储器，关掉地址锁存器门控信号（LDAR=0），关掉数据开关三态门（SW-B=1），片选存储器，使它处于读状态（CE=0，WE=0），此时数据总线上显示的数据即为从存储器当前地址中读出的数据内容。读其他单元依次循环上述步骤。

读存储器操作流程如图所示（以从 00 号单元读出 11H 数据为例）

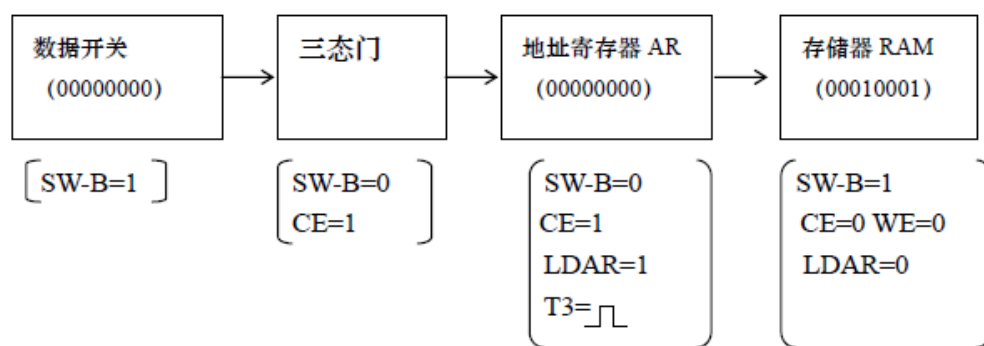


图 18: 读存储器流程

3.4 运行界面

3.5 课程设计总结

通过本次实验，我获得了以下几方面的心得体会：

首先，我对逻辑器件的组成结构有了更深入的了解。实验使我熟悉了一些逻辑器件的工作原理，并验证了它们的组合功能。通过亲自动手进行实验，我更好地理解逻辑器件的构成和功能。

其次，我学会了总线和各个器件之间的工作过程。在实验中，我了解了总线在连接各个器件时的重要作用，以及它们之间的数据传输过程。这让我对整个系统的运作有了更清晰的认识。

同时，实验中遇到了一些问题，但我成功地解决了它们。这个过程让我对所学知识更加牢固，提高了自己解决问题的能力。我也意识到在实验连线时需要细心，并且在操作过程中出现问题时，要能够快速检查并解决，以确保实验顺利进行。

这次实验是在虚拟平台上进行的，我遇到了灯泡不亮的问题，导致我不确定实验是否成功。我及时向老师反映了问题。这次经历让我明白了沟通与求助的重要性。另外，在连接各个器件时，我也注意到了注重美观和整洁的连接方式。

综上所述，通过本次实验，我对逻辑器件的组成和工作原理有了更深入的了解，掌握了可靠的静态存储器的工作特性和使用方法。我通过解决问题提高了自己的知识储备，并且意识到了良好的沟通和整洁的实验操作对于实验成功的重要性。

3.6 参考文献

《计算机组成原理（第二版）》唐朔飞，李国杰，李国杰清华大学出版社

4 题目四

4.1 题目描述

题目：乘法器设计

基本要求：学习乘法器结构，逻辑控制单元设计方法，提出设计方案，小组讨论研究，实现设计方案。

4.2 实验原理

两位二进制数乘法器是一种简单的数字电路设计，用于执行两个二进制数的乘法运算。下面是一个基本的两位二进制数乘法器的设计：

1. 输入：设计一个两位二进制数乘法器，需要两个两位二进制数作为输入。假设这两个数分别为 A 和 B。
2. 乘法运算：将 A 的每一位与 B 的每一位相乘，得到四个部分乘积。对于两位数的乘法，共有四个部分乘积需要计算。
3. 部分乘积对齐：对于第 i 个部分乘积，将其左移 i 位，以对齐相应的位数。
4. 部分乘积相加：将所有四个部分乘积相加，得到最终的乘积结果。
5. 结果输出：输出最终的乘积结果，其位数为输入位数的两倍。

以下是一个具体的示例，展示了两两位二进制数乘法器的设计过程：

输入：A = a₁a₀, B = b₁b₀

1. 计算部分乘积：

$$P_0 = a_0 * b_0$$

$$P_1 = a_0 * b_1$$

$$P_2 = a_1 * b_0$$

$$P_3 = a_1 * b_1$$

2. 对齐部分乘积:

$$P0 = P0 \ll 0 \text{ (不需要对齐)}$$

$$P1 = P1 \ll 1 \text{ (左移 1 位)}$$

$$P2 = P2 \ll 1 \text{ (左移 1 位)}$$

$$P3 = P3 \ll 2 \text{ (左移 2 位)}$$

3. 相加部分乘积:

$$\text{Result} = P0 + P1 + P2 + P3$$

4. 结果输出:

输出 Result 作为两位二进制数的乘积结果。

需要注意的是,这只是一个基本的两位二进制数乘法器的设计示例。在实际应用中,可能要考虑更多的优化和细节,如进位处理、结果截断、错误处理等。此外,也可以使用更复杂的乘法器结构来提高性能和效率。

4.3 实验设计

实验中采用两块 74LS138 芯片，其中一位数值进行片选信号，每一块芯片各自表示 8 个数值，因此可以表示 16 个数值，在数电设计中，列出真值表就可以确定逻辑电路的设计，如下表所示：

表 2: 74LS138 芯片真值表

A1	A0	B1	B0	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

由真值表可以获得以下的逻辑表达式：

$$Y3=m15$$

$$Y2=m10+m11+m14$$

$$Y1=m6+m7+m9+m11+m13+m14$$

$$Y0=m5+m7+m13+m15$$

通过与非门的组合可以得到两位乘法器

4.4 运行界面

4.5 课程设计总结

通过本次实验，我获得了以下几方面的心得体会：

在乘法器的设计方面，我首先通过查阅资料，了解了乘法器的基本原理，然后通过真值表的列出，得到了逻辑表达式，最后通过与非门的组合，得到了两位乘法器的电路图。在实验过程中，我遇到了一些问题，但我成功地解决了它们。这个过程让我对所学知识更加牢固，提高了自己解决问题的能力。我也意识到在实验连线时需要细心，并且在操作过程中出现问题时，要能够快速检查并解决，以确保实验顺利进行。

在实验电路连接过程中，由于与非门的输入端限制了两个的个数，因此我需要将两个与非门的输出端连接到一个与非门的输入。

综上所述，通过本次实验，我对乘法器的组成和工作原理有了更深入的了解，掌握了可靠的乘法器的工作特性和使用方法。我通过解决问题提高了自己的知识储备，并且意识到了良好的沟通和整洁的实验操作对于实验成功的重要性。

4.6 参考文献

《计算机组成原理（第二版）》唐朔飞，李国杰，李国杰清华大学出版社

5 整体总结

在过去的汇编课程设计实验中，我积极参与实验并从中获得了许多宝贵的经验。相比课堂上的学习，实验为我提供了更广阔的学习空间，让我接触到了许多课本上无法涉及的实际应用场景。

首先，通过实验，我学到了许多新知识和技能。我深入了解了相关概念和理论，并通过实际操作将其应用于实验中。这种实践学习方式加深了我对课程内容的理解，使我能够更好地将理论知识与实际问题相结合。

其次，实验不仅加强了我的基础知识，还提高了我的动手能力。在实验过程中，我需要亲自操作连线，并解决实验中出现的问题。这锻炼了我的实践技能和解决问题的能力，使我更加熟练地运用所学的知识。

最重要的是，我非常庆幸我的努力没有白费。我始终坚持不断地寻找问题并积极解决它们，这使我能够成功地完成课设，并达到了预期的目标。这种自我发现问题和解决问题的过程让我更加深入地理解了实验的意义和目的，并培养了我的自学能力和独立思考能力。

综上所述，课设实验是一次宝贵的学习机会，它为我提供了迈出课堂、深入实践的机会。通过实验，我不仅学到了更多的知识，巩固了基础，提高了动手能力，还培养了自我发现问题和解决问题的能力。我相信这些经历将对我未来的学习和职业发展产生积极的影响。

6 附录

6.1 题目一源代码

```
dataarea segment
2  fname db 'test.txt',0
   string db 'input the file name:$'
4  filename db 14,0,14 dup(?)
   username db 'username:',0ah,0dh,'$'
6  password db 'password:',0ah,0dh,'$'
   user db 'JIANG'
8  pass db '123456'
   tempname db 15,?,15 dup(?)
10 countname db $-tempname-02h,'$'
   temppassword db 15,?,15 dup (?)
12 countpassword db $-temppassword-02h
   wrong1 db 'wrong username!',0ah,0dh,'$'
14 wrong2 db 'wrong password!',0ah,0dh,'$'
   login db 'Login Success!',0ah,0dh,'$'
16 stars db '*****'
   ',0ah,0dh,'$'
   beginning db '*****Welcome to the Counting Characters
   System*****',0ah,0dh,'$'
18 starting db 'Do you want to count characters in the file?(Y/N)
   ', '$'
   ending db 'Thank you for using the counting characters
   system',0ah,0dh,'$'
20 success db 'Counting success!$',0ah,0dh,'$'
   error1 db 'Error opening file!',07h,0
22 error2 db 'Error reading file!',07h,0
   string1 db 'number of $'
24 string2 db ':$'
   array db 26 dup(0)
26 others db 0
   buffer db ?
```



```

28     eof db 032h
dataarea ends
30
codes segment
32     assume cs:codes,ds:dataarea
start:
34     mov ax,dataarea;初始化数据段寄存器
        mov ds,ax
36
        lea dx,stars
38     mov ah,09h
        int 21h
40
        lea dx,beginning;显示欢迎信息
42     mov ah,09h
        int 21h
44
        lea dx,stars
46     mov ah,09h
        int 21h
48
input:
50     lea dx,username
        mov ah,09h
52     int 21h

54     lea dx,tempname
        mov ah,0ah
56     int 21h

58     cmp byte ptr tempname+1,05h
        jnz repeat1
60
        mov cx,5
62     mov si,offset user

```

```

        mov di,offset tempname+2
64      mov ax,dataarea
        mov es,ax
66      cld
        repe cmpsb
68      jnz repeat1

        mov dx,offset tempname+2    ;显示输入的字符串
70      mov byte ptr tempname[7], '$'
72      call dosshow

        lea dx,password
74      mov ah,09h
76      int 21h

        lea dx,temppassword
78      mov ah,0ah
80      int 21h

        cmp byte ptr temppassword+1,06h
82      jnz repeat2

84
        mov cx,6
86      mov si,offset pass
        mov di,offset temppassword+2
88      mov ax,dataarea
        mov es,ax
90      cld
        repe cmpsb
92      jnz repeat2

94      mov dx,offset temppassword+2
        mov byte ptr temppassword[8], '$'
96      call dosshow

```

```

98      jmp loginsuccess

100     repeat1:
101         lea dx,wrong1
102         mov ah,09h
103         int 21h
104         jmp input

106     repeat2:
107         lea dx,wrong2
108         mov ah,09h
109         int 21h
110         jmp input

112     loginsuccess:
113         lea dx,login
114         mov ah,09h
115         int 21h
116
118     ;输入是否统计字符
119     request:
120         lea dx,starting;显示提示信息
121         mov ah,09h
122         int 21h
123
124         mov ah,01h;获取键盘输入
125         int 21h
126
127         cmp al,'N';判断是否统计字符
128         je finish;不统计字符，结束程序
129         cmp al,'n'
130         je finish
131
132         cmp al,'Y';统计字符

```

```

        je continue
134      cmp al,'y'
        je continue

136
        mov dl,0ah;回车换行
138      mov ah,2
        int 21h
140      mov dl,0dh
        mov ah,2
142      int 21h

144      jmp request;输入错误，重新输入

146      finish::结束程序
        mov dl,0ah
148      mov ah,2
        int 21h
150      mov dl,0dh
        mov ah,2
152      int 21h

154      lea dx,ending;显示结束信息
        mov ah,09h
156      int 21h

158      mov ah,07h
        int 21h
160      mov ax,4c00h
        int 21h
162      continue:
        mov dl,0ah
164      mov ah,2
        int 21h
166      mov dl,0dh
        mov ah,2

```

```

168         int 21h

170         lea dx,string;显示提示信息
171         mov ah,09h
172         int 21h

174
175         ;输入文件名
176         lea dx,filename
177         mov ah,0ah
178         int 21h

180         mov bl,filename+1;获取文件名长度
181         mov bh,0
182         mov [bx+filename+2],0;文件名后加0
183         lea dx,filename+2
184         mov ax,3d00h;打开文件
185         int 21h

186
187         ; mov dx,offset fname;打开文件
188         ; mov ah,3dh
189         ; mov al,0
190         ; int 21h

192
193     jnc open
194         mov si,offset error1;打开文件失败
195         call dmess
196         jmp continue

198     open:
199         mov bx,ax
200     go:
201         call readchar;读取一个字符
202         jc readerror;读取失败

```

```

204         cmp al, eof; 判断是否到文件尾
           jz typeok; 到文件尾, 显示结果
           call punch; 统计字符
206         jmp go; 继续读取

208     readererror:
           mov si, offset error2
210         call dmess

212
           typeok:
214         mov ah, 3eh; 关闭文件
           int 21h
216         mov dl, 0ah
           mov ah, 2
218         int 21h
           call show

220
           ; 回车换行
           mov dl, 0ah
224         mov ah, 2
           int 21h
226         mov dl, 0dh
           mov ah, 2
228         int 21h

           lea dx, success; 显示成功信息
230         mov ah, 09h
           int 21h
232
234     mov dl, 0ah
           mov ah, 2
236         int 21h
           mov dl, 0dh

```

```

238      mov ah,2
      int 21h

240
      jmp request;重新输入

242

244 over:
      lea dx,ending;显示结束信息
246      mov ah,09h
      int 21h

248
      mov ah,07h
250      int 21h
      mov ax,4c00h
252      int 21h

254
;子程序段
256 ;读取一个字符
readchar proc
258      mov cx,1
      mov dx,offset buffer
260      mov ah,3fh
      int 21h
262      jc r1
      cmp ax,cx
264      mov al,eof
      jb r2
266      mov al,buffer

268 r2:
      clc

270 r1:
      ret
272 readchar endp

```

```

274 ;显示错误信息
    dmess proc
276 dmess1:
        mov dl,[si]
278        inc si
        or dl,dl
280        jz dmess2
        mov ah,02h
282        int 21h
        jmp dmess1
284 dmess2:
        ret
286 dmess endp

288 ;统计字符
    punch proc
290        push dx
        mov dl,al
292        mov ah,02h
        int 21h
294        pop dx
        mov cl,41h
296        lea di,array
        mov ch,al
298        cmp ch,cl
        jb other
300        cmp ch,5ah
        ja higher2
302
    ;统计大写字母
304 h1:
        je char
306        ja loop1

```



```

308     loop1:
310         inc cl
312         add di,1
314         jmp h1
316
318     ;统计小写字母
320     higher2:
322         mov cl,61h
324         lea di,array
326         mov ch,al
328         cmp ch,cl
330         jb other
332         cmp ch,7ah
334         ja other
336
338     h2:
340         cmp ch,cl
342         je char
344         ja loop2
346
348     loop2:
350         inc cl
352         add di,1
354         jmp h2
356
358     char:
360         sub ch,ch
362         mov ch,[di]
364         inc ch
366         mov [di],ch
368
370     other:
372         inc others
374
376     ret

```

```

344      punch endp

346      ;显示统计结果
      show proc
348          lea si,array
          mov di,41h
350      loop3:
          lea dx,string1
352          mov ah,09h
          int 21h
354          mov dx,di
          mov ah,02h
356          int 21h
          lea dx,string2
358          mov ah,09h
          int 21h
360          sub ax,ax
          mov al,[si]
362          add si,1
          call display
364          call endlne
          inc di
366          cmp di,5bh
          jb loop3
368          ret
      show endp

370
      endlne proc near
372      mov dl,20h
          mov ah,02h
374          int 21h
          mov dl,20h
          mov ah,02h
376          int 21h

```

```

378         mov dl,20h
           mov ah,02h
380         int 21h
           mov dl,20h
382         mov ah,02h
           int 21h
384         mov dl,20h
           mov ah,02h
386         int 21h
           ret
388 endline endp

390 ;显示一个字节
display proc near
392     mov bl,10
           div bl
394     push ax
           mov dl,al
396     add dl,30h
           mov ah,02h
398     int 21h
           pop ax
400     mov dl,ah
           add dl,30h
402     mov ah,02h
           int 21h
404     mov dl,20h
           mov ah,02h
406     int 21h
           ret
408 display endp

410 ;显示字符串
dosshow proc
412     mov ah,09h

```

```

    int 21h
414
    mov dl,0dh
416    mov ah,02h
    int 21h
418
    mov dl,0ah
420    mov ah,02h
    int 21h
422
    ret
424 dosshow endp
426 codes ends
end start
```

题目一源代码

6.2 题目二源代码

```
1      DATAS SEGMENT
STR1  DB  'n=', '$';定义提示字符
3 STR2  DB  'n!=', '$';定义字符，显示结果
MSG0   DB  '
          *****'
          ,0AH,0DH, '$'
5 MSG1   DB  '*****Welcome to the Calculating N
          Factorial Program*****',0AH,0DH, '$'
MSG2   DB  'Do you want to Calculate N Factorial?(Y/N)', '$'
7 ending db 'Thank you for using the Calculating N Factorial
          Program',0ah,0dh, '$'
username db 'username:',0ah,0dh, '$'
9 password db 'password:',0ah,0dh, '$'
user db 'JIANG'
11 pass db '123456'
tempname db 15,?,15 dup(?)
13 countname db $-tempname-02h, '$'
temppassword db 15,?,15 dup (?)
15 countpassword db $-temppassword-02h
wrong1 db 'wrong username!',0ah,0dh, '$'
17 wrong2 db 'wrong password!',0ah,0dh, '$'
login db 'Login Success!',0ah,0dh, '$'
19
ANS DW  1,3000 DUP(-1)      ;储存运算结果 存入一个1应对输入0的情
况
21 ANSH  DW  3000 DUP(0)      ;相对高位
ANSL  DW  3000 DUP(0)      ;相对低位
23 DATAS ENDS

25 CODES SEGMENT
ASSUME  CS:CODES,DS:DATAS
27
;字符输出
```

```

29 OUTPUTCHAR MACRO AINCHAR    ;将字符AINCHAR输出
    PUSH AX
31    PUSH BX
    PUSH CX
33    PUSH DX

    MOV DL,AINCHAR
35    MOV AH,02H                ;输出字符
    INT 21H
37

    POP DX
    POP CX
39    POP BX
    POP AX
41
43 ENDM

45 ;字符串输出
OUTPUTSTR MACRO AIMSTR        ;将字符串AIMSTR输出
47    PUSH AX
    PUSH BX
49    PUSH CX
    PUSH DX

51    LEA DX,AIMSTR              ;将AIMSTR的偏移地址送到DX寄存器
53    MOV AH,09H                ;09H字符串输出功能
    INT 21H

55    POP DX
    POP CX
57    POP BX
    POP AX
59
61 ENDM

    ;以10进制输出AX中的数值
63 OUTPUTAX MACRO              ;将AX中的数值以10进制形式输出

```

```

        PUSH AX
65     PUSH BX
        PUSH CX
67     PUSH DX
        CALL OUTPUTAXP      ;调用进制输出过程
69     POP DX
        POP CX
71     POP BX
        POP AX
73 ENDM

75 OUTPUTAXP PROC
    MOV DX,0
77     MOV CX,0              ;用CX储存余数个数后续LOOP需要使用
        CMP AX,0            ;判断AX中的值是否为0
79     JNE OUTPUTAXF1
        OUTPUTCHAR '0'
81     JMP OUTPUTAXPEXIT

83 OUTPUTAXF1:
    CMP AX,0                ;判断AX中的值是否为0
85     JE OUTPUTAXF2         ;是则说明AX已经按位除完了
    MOV BX,10                ;10进制
87     DIV BX                ;除10
        PUSH DX              ;将余数入栈保存
89     MOV DX,0
    INC CX                  ;计数循环取得的余数个数
91     JMP OUTPUTAXF1

93 OUTPUTAXF2:              ;循环输出取得的余数
    POP AX
95     ADD AL,30H
        OUTPUTCHAR AL
97     LOOP OUTPUTAXF2
OUTPUTAXPEXIT: RET

```

```

99 OUTPUTAXP ENDP

101 ;输出字符串AIMNUM所表示的数值
    OUIPUTNUM MACRO AIMNUM
103     PUSH AX
        PUSH BX
105     PUSH CX
        PUSH DX
107     PUSH SI

109     LEA BX,AIMNUM;用BX存储字符串AIMNUM在DS中的首地址
        CALL OUTPUTNUMP ;调用字符串AIMNUM数值输出过程

111
        POP SI
113     POP DX
        POP CX
115     POP BX
        POP AX
117 ENDM

119 OUTPUTNUMP PROC
    OUTPUTNUMF1:
121     MOV SI,-2
    OUTPUTNUMEND: ;使SI指向ANS的数值结尾处
123     ADD SI,2
        MOV AX,[BX+SI] ;测试AX是否为-1
125     CMP AX,-1
        JNE OUTPUTNUMEND ;直到搜索到最后结尾-1
127
        SUB SI,2
129     CMP SI,-2
        JE OUTPUTNUMEXIT ;若为-2则说明ANS中不存在数据
131     MOV AX,[BX+SI] ;取出ANS中的第一个数值到AX中 从低到高
        OUTPUTAX ;将AX中的数以10进制形式输出 是最高位不需要填
            0

```



```

133 OUTPUTNUMNEXT:
135     SUB SI,2
      CMP SI,-2
137     JE OUTPUTNUMEXIT
      MOV AX,[BX+SI]           ;取出ANS中的数值到AX中 开始判断有多少0需
                               要填充
139     CMP AX,1000
      JAE OUTPUTNUMF2         ;AX中的数值大于等于1000时跳转
141     OUTPUTCHAR '0'         ;AX小于1000时先输出一个字符'0'
      CMP AX,100
143     JAE OUTPUTNUMF2
      OUTPUTCHAR '0'         ;AX小于100时再输出一个字符'0'
145     CMP AX,10
      JAE OUTPUTNUMF2
147     OUTPUTCHAR '0'         ;AX小于10时再输出一个字符'0'

149 OUTPUTNUMF2:
      OUTPUTAX                ;将AX中的数以10进制形式输出
151     JMP OUTPUTNUMNEXT     ;跳转进行下一位数值的输出
      OUTPUTNUMEXIT:
153     RET
      OUTPUTNUMP ENDP

155
157
157 START:
159     MOV AX,DATAS
      MOV DS,AX

161     OUTPUTSTR MSG0
163     OUTPUTSTR MSG1
      OUTPUTSTR MSG0

165     input:

```

```

167         lea dx,username
168         mov ah,09h
169         int 21h

171         lea dx,tempname
172         mov ah,0ah
173         int 21h

175         cmp byte ptr tempname+1,05h
176         jnz repeat1

177         mov cx,5
178         mov si,offset user
179         mov di,offset tempname+2
180         mov ax,DATAS
181         mov es,ax
182         cld
183         repe cmpsb
184         jnz repeat1

187         mov dx,offset tempname+2 ;显示输入的字符串
188         mov byte ptr tempname[7], '$'
189         call dosshow

191         lea dx,password
192         mov ah,09h
193         int 21h

195         lea dx,temppassword
196         mov ah,0ah
197         int 21h

199         cmp byte ptr temppassword+1,06h
200         jnz repeat2
201

```

```

        mov cx,6
203    mov si,offset pass
        mov di,offset temppassword+2
205    mov ax,DATAS
        mov es,ax
207    cld
        repe cmpsb
209    jnz repeat2

211    mov dx,offset temppassword+2
        mov byte ptr temppassword[8], '$'
213    call dosshow

215    jmp loginsuccess

217    repeat1:
        lea dx,wrong1
219    mov ah,09h
        int 21h
221    jmp input

223    repeat2:
        lea dx,wrong2
225    mov ah,09h
        int 21h
227    jmp input

229    loginsuccess:
        lea dx,login
231    mov ah,09h
        int 21h
233
request:
235    lea dx,MSG2;显示提示信息
        mov ah,09h

```

```

237         int 21h

239     mov ah,01h;获取键盘输入
        int 21h

241

        cmp al,'N';判断是否统计字符
243     je finish;不统计字符，结束程序
        cmp al,'n'
245     je finish

        cmp al,'Y';统计字符
247     je continue
        cmp al,'y'
249     je continue

251

        mov dl,0ah
253     mov ah,2
        int 21h
255     mov dl,0dh
        mov ah,2
257     int 21h

259     jmp request;输入错误，重新输入

261 finish:
        mov dl,0ah
263     mov ah,2
        int 21h
265     mov dl,0dh
        mov ah,2
267     int 21h

269     lea dx,ending;显示结束信息
        mov ah,09h
271     int 21h

```

```

273         mov ah,07h
           int 21h
275         mov ax,4c00h
           int 21h
277     continue:
        mov dl,0ah
279         mov ah,2
           int 21h
281         mov dl,0dh
           mov ah,2
283         int 21h

285     OUTPUTSTR STR1      ;输出字符串STR1
        PUSH BX
287     PUSH CX
        PUSH DX
289
        MOV AX, 0
291 TYPEIN:  ;输入要求解的n值
        PUSH AX
293     MOV AH,01H
        INT 21H                ;字符默认输入到AL中
295     CMP AL,13
        JE TYPEINEXIT          ;检测到回车后跳转AX的输出
297     SUB AL,48                ;将字符转化为对应的数值
        MOV BH,0
299     MOV BL,AL
        POP AX
301     CMP AX,0                ;当AX中的数值为0时，跳过乘法操作
        JE TYPEINADD
303     MOV CX,10
        MUL CX                  ;乘以10
305 TYPEINADD:
        ADD AX,BX

```

```

307     JMP TYPEIN
TYPEINEXIT:
309     POP AX                ;将计算得到的数值出栈到AX中
        POP DX
311     POP CX
        POP BX
313     MOV CX,AX            ;求阶乘的数转至CX中
;输入结束
315     OUTPUTSTR STR2      ;输出字符串STR2

317
;计算阶乘并保存到ANS
319     MOV BX,1            ;BX逐步求阶的乘数
SAVENEXT:
321     CMP CX,0
        JE  OUTPUTANS      ;当CX中的值为0时，输出ANS中的数值
323     PUSH CX
        MOV SI,0           ;SI指向ANS的起始位置

325
MULANS:      ;对ANS中的所有数值进行乘BX操作，乘积大于等于
            10000的部分存储到ANSH中，小于10000的部分存储到ANSL中
327     MOV AX,ANS[SI]      ;取出ANS中的数值到AX中
        CMP AX,-1
329     JE  TRANSFORM      ;直到取得的数值为-1时，跳转
        MUL BX            ;进行乘法操作

331
        PUSH CX
333     MOV CX,10000
        DIV CX            ;除法操作 除以10000
335     POP CX

337     MOV ANSL[SI],DX     ;将余数存储到ANSL中
        ADD SI,2
339     MOV ANSH[SI],AX     ;将商存储到ANSH中

```

```

341     JMP MULANS

343 TRANSFORM:                ;对ANS乘以BX得到的数值字符串ANSL和ANSH，进
    行格式调整，并将调整后的结果存储到ANS中去
    PUSH BX                  ;BX中的乘数入栈保存
345     MOV BX,0
    MOV SI,2
347
    TRANSFORMF1:
349     MOV AX,ANS[BX]        ;取出ANS中的数值到AX中
    CMP AX,-1
351     JE  TRANSFORMF2      ;当ANS中的数值取完时，跳转

353     MOV AX,ANSH[BX]      ;取商到AX中
    ADD AX,ANSL[BX]         ;加上此时所在位置对应的余数
355
    CMP AX,10000            ;判断AX中的数值是否大于10000
357     JB  SAVEINTOANS      ;小于10000时直接将数值存储到ANS中
    MOV DX,0                ;大于10000时，将大于等于10000的部分存到高
    位的进位中去，小于10000的部分存储到ANS中
359     PUSH CX
    MOV CX,10000
361     DIV CX
    POP CX
363     MOV ANS[BX],DX        ;小于10000的余数部分存储到ANS中
    ADD ANSH[SI],AX         ;大于10000的高位部分添加到高位的进位中去
365
    ADD BX,2                ;指针后移指向下一个数值
367     ADD SI,2
    JMP TRANSFORMF1
369
    SAVEINTOANS:
371     MOV ANS[BX],AX        ;将数值存储到ANS中
    ADD BX,2                ;指针后移指向下一个数值
373     ADD SI,2

```

```

375     JMP TRANSFORMF1

TRANSFORMF2:
377     MOV AX,ANSH[BX]      ;取出上一个商到AX中
        CMP AX,0
379     JE TRANSFORMF3      ;若AX中的数值为0时 跳过下一步
        MOV ANS[BX],AX      ;将上一位的商添加到ANS中
381 TRANSFORMF3:
        POP BX              ;BX中的数值出栈
383     INC BX
        POP CX
385     LOOP SAVENEXT

387 OUTPUTANS:              ;输出数值字符串所表示的数值
        OUTPUTNUM ANS

389
        mov dl,0ah
391     mov ah,2
        int 21h
393     mov dl,0dh
        mov ah,2
395     int 21h

397     jmp request

399     MOV AH,4CH
        INT 21H

401
        dosshow proc
403     mov ah,09h
        int 21h

405
        mov dl,0dh
407     mov ah,02h
        int 21h

```



```
409      mov dl,0ah
411      mov ah,02h
      int 21h
413
      ret
415 dosshow endp
CODES ENDS
417 END START
```

题目二源代码