

科技学院

**课程设计(综合实验)报告**

( 2019-- 2020年度第 2 学期)

名 称： 汇编语言课程设计

题 目： 汇编语言实践综合

院 系： 计算机系

班 级： 软件1802

学 号： 201809000121

学生姓名： 王毅涛

指导教师： 刘书刚

设计周数： 一 周（分散进行）

成 绩：

日期：2020年 7月 5 日

导航页

## [实验一](#实验一)

## [实验二](#实验二)

## [实验三](#实验三)

## [实验四](#实验四)

# 汇编语言基本认识

**一、课程设计(综合实验)的目的与要求**

通过本门课程的学习与实践，学生可以加深对汇编语言程序设计课程的理解与掌握，有助于提高学生的汇编语言程序设计能力，同时可以加深对计算机工作原理的理解，有助于促进后续课程的学习。

本课程要求学生针对课本所学知识从调试程序、循环程序设计、分支程序设计、子程序设计、键盘输入程序设计和中断练习程序等部分学习实践，使学生对汇编语言有更深入细致和全面的了解，并能用所学知识解决实际问题，每个实验的具体要求如下：

各个实验要求有设计说明，对有实现要求的程序画出程序流程图，然后根据流程图编写程序，并上机调试写出运行结果。

**二、设计（实验）正文**

（1）设计思路

使用EMU8086软件，逐个键入命令行，然后用EMULATE功能做仿真

（2）程序框图

略

（3）程序实现

（1）MOV AX,0102H

（2）MOV AL,05H

（3）MOV AX,1234H

（4）MOV BX,AX

（5）MOV AH,BX；对吗？环境如何报错？为什么？

（6）MOV AX,[2000H]

;执行前2000H单元放置的数据是什么？执行后是不是放在了目的操作数中？执行前是否可以给2000H单元放置数据？怎么放？

（7）MOV AX,ES:[2000H]

; 和上条的区别？尝试给内存单元先赋值，然后再执行这条指令

（8）MOV [2000H]，AX

（9）MOV [2000H]，[3000H]；可行吗？环境如何报错？为什么指令错误？

（10）MOV AX,[BX]

；先给bx赋值，然后查看相应内存单元的内容，最后执行这条指令，是否达到要求？

（11）完成下述类似功能：设法给数据段用BX指示的一个字赋值为1000H，给堆栈段BP指示的一个字赋值为2335H，则下列指令执行后AX寄存器的内容是多少？

MOV AX，[BX]

MOV AX，[BP]

MOV AX，DS：[BP]

（12）MOV AX,[DI+06H] ；事先给相应寄存器、内存单元赋值

（13）MOV AX,[BP+06H ；事先给相应寄存器、内存单元赋值

（14）MOV AX,[BX+SI] ；事先给相应寄存器、内存单元赋值

（15）MOV AX,[BP+DI] ；事先给相应寄存器、内存单元赋值

（16）MOV AX,DS:[BP+DI] ；事先给相应寄存器、内存单元赋值

（17）MOV AX,[BX+SI+06H] ；事先给相应寄存器、内存单元赋值

（18）设法观察一下具体指令占有的内存大小。

（4）实验结果

（1）MOV AX,0102H

（2）MOV AL,05H

（3）MOV AX,1234H

（4）MOV BX,AX

正确执行无误。

（5）MOV AH,BX；对吗？环境如何报错？为什么？

报错为WRONG PARAMETERS(错误参数)，实际上这是把不同位数的寄存器片段强行互相赋值的必然结果。

（6）MOV AX,[2000H]

;执行前2000H单元放置的数据是什么？执行后是不是放在了目的操作数中？执行前是否可以给2000H单元放置数据？怎么放？

AX=1234,然后是AX=0000.是的;可以使用MOV [2000H],2468H

（7）MOV AX,ES:[2000H]

; 和上条的区别？尝试给内存单元先赋值，然后再执行这条指令

段基址寄存器不一样。先执行MOV ES:[2000H],1919H，然后AX就会存放1919H

（8）MOV [2000H]，AX

（9）MOV [2000H]，[3000H]；可行吗？环境如何报错？为什么指令错误？

不可行。报错WRONG PARAMETERS。因为MOV指令的两个参数不可以同时为内存地址。

（10）MOV AX,[BX]

；先给bx赋值，然后查看相应内存单元的内容，最后执行这条指令，是否达到要求？

AX最终存放的是0000H，实际上，这是导向了主存地址里0012的存放的数据。此处既然没有初始化，那么就是0000

（11）完成下述类似功能：设法给数据段用BX指示的一个字赋值为1000H，给堆栈段BP指示的一个字赋值为2335H，则下列指令执行后AX寄存器的内容是多少？

MOV AX，[BX]

MOV AX，[BP]

MOV AX，DS：[BP]

AX此处显示为2335H

（12）MOV AX,[DI+06H] ；事先给相应寄存器、内存单元赋值

MOV DI,8H

MOV [DI+06H],50H

MOV AX,[DI+06H]

AX执行完成后为0050h

（13）MOV AX,[BP+06H ；事先给相应寄存器、内存单元赋值

同12

（14）MOV AX,[BX+SI] ；事先给相应寄存器、内存单元赋值

同12，另外赋值

（15）MOV AX,[BP+DI] ；事先给相应寄存器、内存单元赋值

同12，另外赋值

（16）MOV AX,DS:[BP+DI] ；事先给相应寄存器、内存单元赋值

同12，另外赋值

（17）MOV AX,[BX+SI+06H] ；事先给相应寄存器、内存单元赋值

同12，另外赋值

（18）设法观察一下具体指令占有的内存大小。

寄存器-立即数赋值占3个单元，看具体情况。

**三、课程设计（综合实验）总结或结论**

实验难度不大。本次实际操作下来，我对于EMU8086软件的使用熟练度提高，加深了汇编语言中寄存器和主存的使用方式的了解。并且通过查阅资料和查阅网络资料的方式了解了一些额外的知识。

**四、参考文献**

《IBM-PC汇编语言程序设计（沉美明第二版）》

# 指令系统实践

**一、课程设计(综合实验)的目的与要求**

通过本门课程的学习与实践，学生可以加深对汇编语言程序设计课程的理解与掌握，有助于提高学生的汇编语言程序设计能力，同时可以加深对计算机工作原理的理解，有助于促进后续课程的学习。

本课程要求学生针对课本所学知识从调试程序、循环程序设计、分支程序设计、子程序设计、键盘输入程序设计和中断练习程序等部分学习实践，使学生对汇编语言有更深入细致和全面的了解，并能用所学知识解决实际问题，每个实验的具体要求如下：

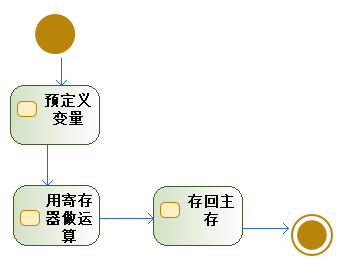
各个实验要求有设计说明，对有实现要求的程序画出程序流程图，然后根据流程图编写程序，并上机调试写出运行结果。

**二、设计（实验）正文**

1. 设计思路

XYZW均是预先定义的变量，我的思路是利用定义变量的方法将其写入主存，再使用寄存器作为暂存手段，一步步累积结果，最后赋值到主存中的目的区域。

1. 程序框图



1. 程序实现

第一部分：

1. data segment
2. x dd 1
3. y dd 2
4. zed dd 3  ;同Z，但是Z是保留字
5. w dd 4
7. pkey db "press any key...$"
8. ends
10. stack segment
11. dw   128  dup(0)
12. ends
14. code segment
15. assume cs:code,ds:data
16. start:
17. ; set segment registers:
18. mov ax, data
19. mov ds,ax
20. mov ax,x
21. ADD AX,Y
22. ADD AX,24
23. SUB AX,zed
24. mov w,ax

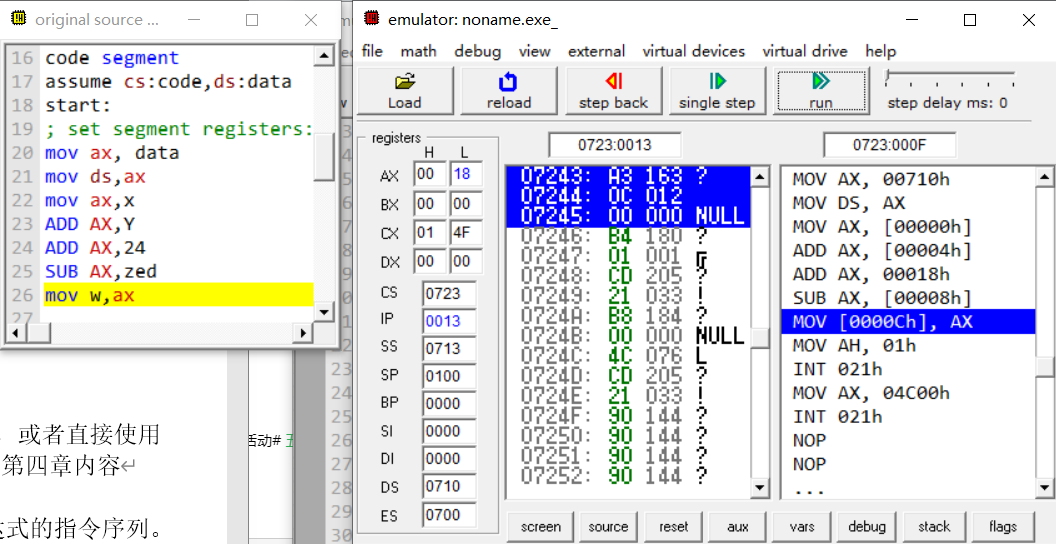


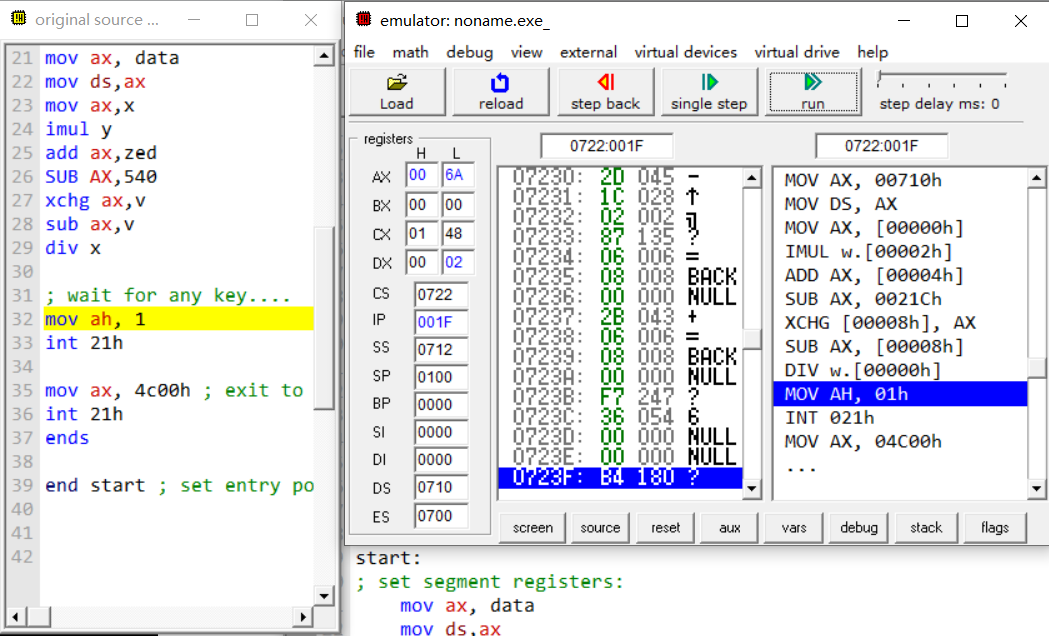
29. ; wait for any key....
30. mov ah, 1
31. int 21h
33. mov ax, 4c00h ; exit to operating system.
34. int 21h
35. ends
37. end start ; set entry point and stop the assembler.

第二部分：

1. ; multi-segment executable file template.
3. data segment
4. x dw 5
5. y dw 2
6. zed dw 3  ;同Z，但是Z是保留字
7. w dw 4
8. v dw 5
10. pkey db "press any key...$"
11. ends
13. stack segment
14. dw   128  dup(0)
15. ends
17. code segment
18. assume cs:code,ds:data
19. start:
20. ; set segment registers:
21. mov ax, data
22. mov ds,ax
23. mov ax,x
24. imul y
25. add ax,zed
26. SUB AX,540
27. xchg ax,v
28. sub ax,v
29. div x
31. ; wait for any key....
32. mov ah, 1
33. int 21h
35. mov ax, 4c00h ; exit to operating system.
36. int 21h
37. ends
39. end start ; set entry point and stop the assembler.

（4）实验结果

效果1：

效果2：

**三、课程设计（综合实验）总结或结论**

本次实验关键在于拆解长运算式，以及寄存器-主存片段位数匹配.一开始没有给段寄存器做初始化，出现了严重问题，后来经过改正之后正常运行。

**四、参考文献**

《IBM-PC汇编语言程序设计（沉美明第二版）》

# 显示ASCII字符表，统计字符数目

**一、课程设计(综合实验)的目的与要求**

通过本门课程的学习与实践，学生可以加深对汇编语言程序设计课程的理解与掌握，有助于提高学生的汇编语言程序设计能力，同时可以加深对计算机工作原理的理解，有助于促进后续课程的学习。

本课程要求学生针对课本所学知识从调试程序、循环程序设计、分支程序设计、子程序设计、键盘输入程序设计和中断练习程序等部分学习实践，使学生对汇编语言有更深入细致和全面的了解，并能用所学知识解决实际问题，每个实验的具体要求如下：

各个实验要求有设计说明，对有实现要求的程序画出程序流程图，然后根据流程图编写程序，并上机调试写出运行结果。

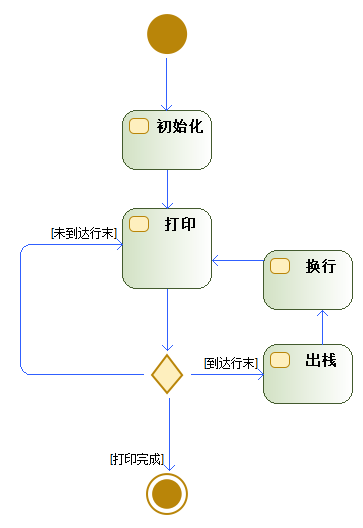
**二、设计（实验）正文**

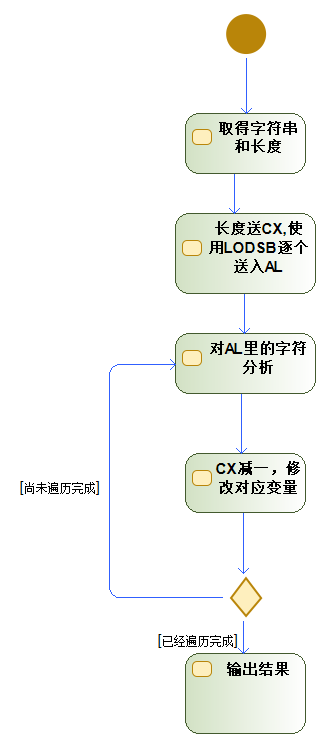
（1）设计思路

用表格形式显示ASCII字符SMASCII

按15行X16列的表格显示表示ASCII码为10H—100H的所有字符，即以行为主的顺序及ASCII码递增的次序一次显示对应的字符。每16个字符为一行，每行中的相邻的字符之间用空白（ASCII为0）隔开。显示空白符时，用其ASCII码置入dl寄存器。每行结束时，用显示回车（ASCII为0dh）和换行符（ASCII为0ah）来结束本行并开始下一行。

（2）程序框图（画出程序主要部分流程图）





（3）程序实现（注意：不用写出整体程序，写出部分关键代码即可，代码后要求用文字描述代码完成的功能）

第一部分：

1. data segment
2. ; add your data here!
3. pkey db "press any key...$"
4. VAR1 DB 00H
5. ends
7. stack segment
8. dw   128  dup(0)
9. ends

12. code segment
13. start:
14. MOV DX,0d0aH
15. PUSH DX ;PUSH IN
16. MOV DL,10H
17. MOV BX,15;行数
18. OUTER:
19. MOV CX,16; 列数
20. ;MOV DL,10H
21. A:
22. MOV AH,02H
23. **INT** 21H
24. INC DL
25. LOOP A
26. ;换行
27. MOV VAR1,DL ;暂存DL
28. pop dx
29. **int** 21h
30. push cx
31. mov cl,8
32. ror dx,cl
33. **int** 21h
34. ror dx,cl
35. pop cx
36. push dx
37. MOV DL,VAR1;暂存DL
38. ;换行
39. DEC BX
40. CMP BX,0
41. JNE OUTER
43. ends
44. end start ; set entry point and stop the assembler.

第二部分：

1. ; multi-segment executable file **template**.
3. data segment
4. ; add your data here!
5. buf 80,?,80 DUP(?)
6. letter db 0
7. digit db 0
8. other db 0
9. ten db 10
10. lstr db "char number==$"
11. dstr db "digit number==$"
12. ostr db "others number==$"
13. pkey db "press any key...$"
14. ends
16. stack segment
17. dw   128  dup(0)
18. ends
20. code segment
21. start:
22. ; set segment registers:
23. mov ax, data
24. mov ds, ax
25. mov es, ax
27. mov cx,0
29. mov dx,offset buf
30. mov ah,0ah
31. **int** 21h
33. mov cl,buf[1]
34. mov si,(offset buf)+2
35. loops:
36. lodsb
37. CMP AL,30H
38. JB COTHER
39. CMP AL,3AH
40. JB CDIGIT
41. CMP AL,41H
42. JB COTHER
43. CMP AL,5AH
44. JBE CLETTER
45. CMP AL,61H
46. JB COTHER
47. CMP AL,7AH
48. JBE CLETTER
49. CLETTER:
50. inc letter
51. jmp HUB
52. COTHER:
53. INC other
54. jmp HUB
55. CDIGIT:
56. inc digit
57. jmp HUB
58. HUB:
59. DEC CX
60. JZ OUTPUT
61. jmp loops

64. OUTPUT:
65. ;NOW HOW TO PRINT THESE CHARS?
66. CALL RELINE
67. lea dx, lstr
68. mov ah,9
69. **int** 21h;×ÖÄ¸Ç°ÖÃ½éÉÜ
70. mov dl,letter
71. MOV AL,DL
72. MOV AH,0
73. CALL PRINT10;´òÓ¡×ÖÄ¸¸öÊý
74. call RELINE
75. lea dx,dstr
76. mov ah,9
77. **int** 21h;´òÓ¡Êý×Ö½éÉÜ
78. mov dl,digit
79. MOV AL,DL
80. MOV AH,0
81. CALL PRINT10;´òÓ¡Êý×Ö¸öÊý
82. call RELINE
83. lea dx,ostr
84. mov ah,9
85. **int** 21h;´òÓ¡ÆäËû½éÉÜ
86. mov dl,other
87. MOV AL,DL
88. MOV AH,0
89. call PRINT10;´òÓ¡ÆäËû¸öÊý
90. CALL RELINE
91. CALL TERMINATE

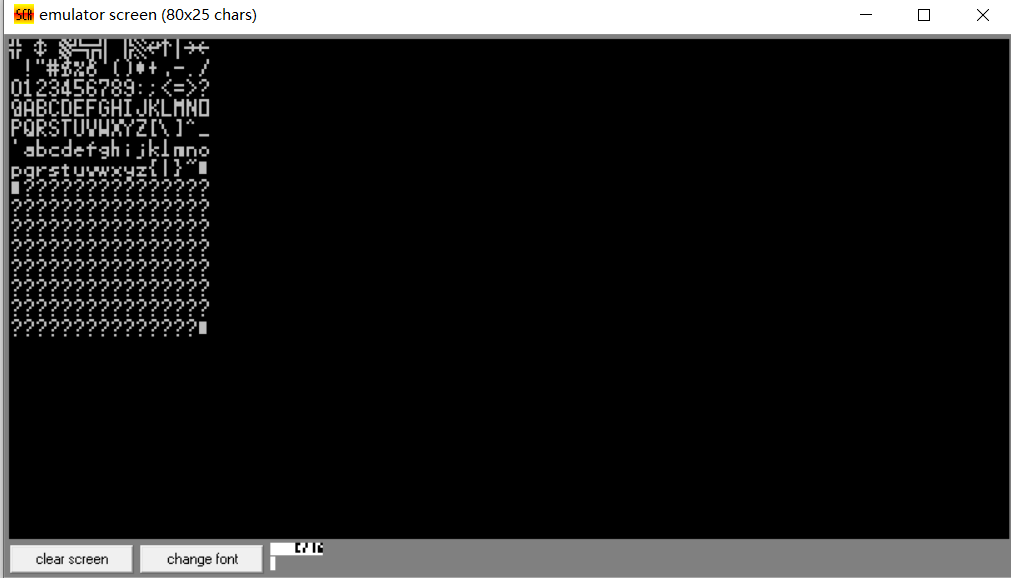
94. RELINE:
95. mov ah,02h
96. mov dl,0dh
97. **int** 21h
98. mov ah,02h
99. mov dl,0ah
100. **int** 21h
101. RET
103. PRINT10:
104. DIV ten
105. CALL PRINTAH
106. CMP AL,0
107. Jne PRINT10
108. RET
110. PRINTAH:
111. PUSH AX
112. PUSH DX
113. MOV DL,AH
114. add dl,48
115. MOV AH,2
116. **INT** 21H
117. POP DX
118. POP AX
119. RET

122. TERMINATE:
123. lea dx, pkey
124. mov ah, 9
125. **int** 21h        ; output string at ds:dx
127. ; wait **for** any key....
128. mov ah, 1
129. **int** 21h
131. mov ax, 4c00h ; exit to operating system.
132. **int** 21h
133. RET

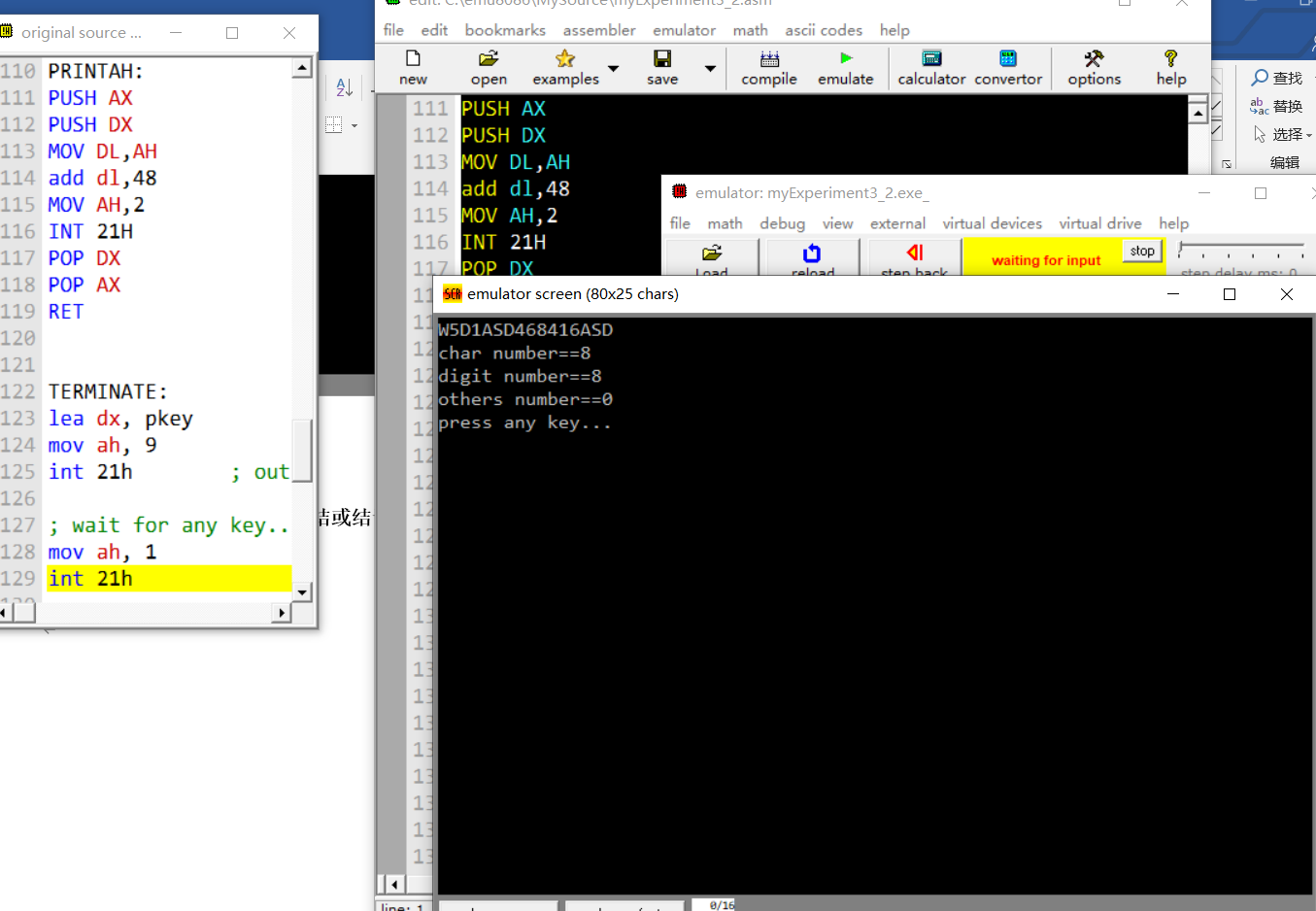
136. ends
138. end start ; set entry point and stop the assembler.

（4）实验结果（实验的运行结果）

第一部分：



第二部分：



**三、课程设计（综合实验）总结或结论**

本次实验是考察具体的串和ASCII码相关操作，当然涉及到了条件跳转等知识。第一个实验部分比较简单，但是由于字体原因，码表没有显示完全，第二个实验部分比较困难，困难在于如何处理串，如何分割串，如何统计，如何输出，以及把放在寄存器里的数字用ASCII转换，还有最重要的，十进制数字输出。我采用的是比较通用的，也就是引入除法，逐个送打印。

**四、参考文献**

《IBM-PC汇编语言程序设计（沉美明第二版）》

# 查找匹配字符串与电话号码本

**一、课程设计(综合实验)的目的与要求**

通过本门课程的学习与实践，学生可以加深对汇编语言程序设计课程的理解与掌握，有助于提高学生的汇编语言程序设计能力，同时可以加深对计算机工作原理的理解，有助于促进后续课程的学习。

本课程要求学生针对课本所学知识从调试程序、循环程序设计、分支程序设计、子程序设计、键盘输入程序设计和中断练习程序等部分学习实践，使学生对汇编语言有更深入细致和全面的了解，并能用所学知识解决实际问题，每个实验的具体要求如下：

各个实验要求有设计说明，对有实现要求的程序画出程序流程图，然后根据流程图编写程序，并上机调试写出运行结果。

**二、设计（实验）正文**

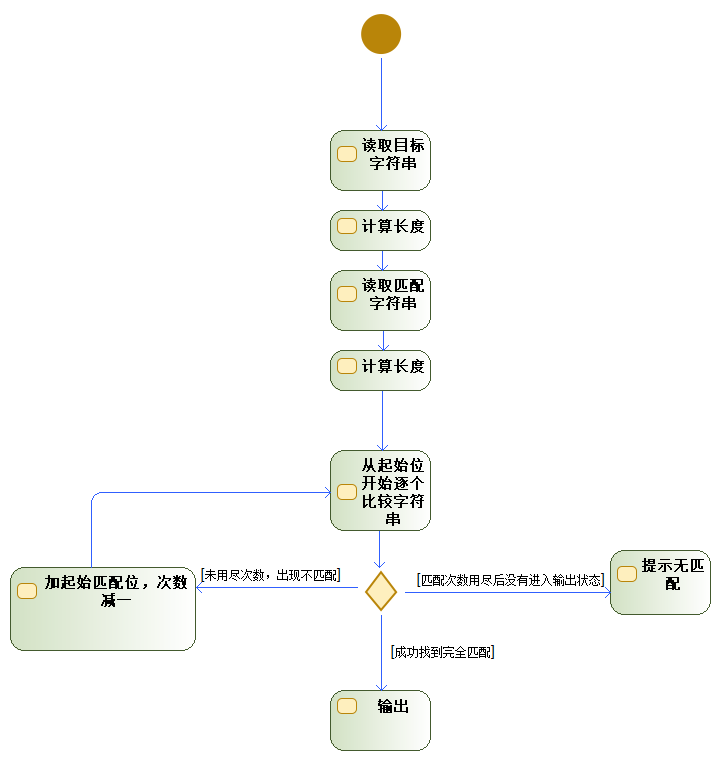
（1）设计思路

实验一：各自读取匹配字符串和目标字符串，得到长度，使用串操作的CMPSB，设定不同的SI,DI,直到成功达成完全匹配，否则就宣告无匹配。

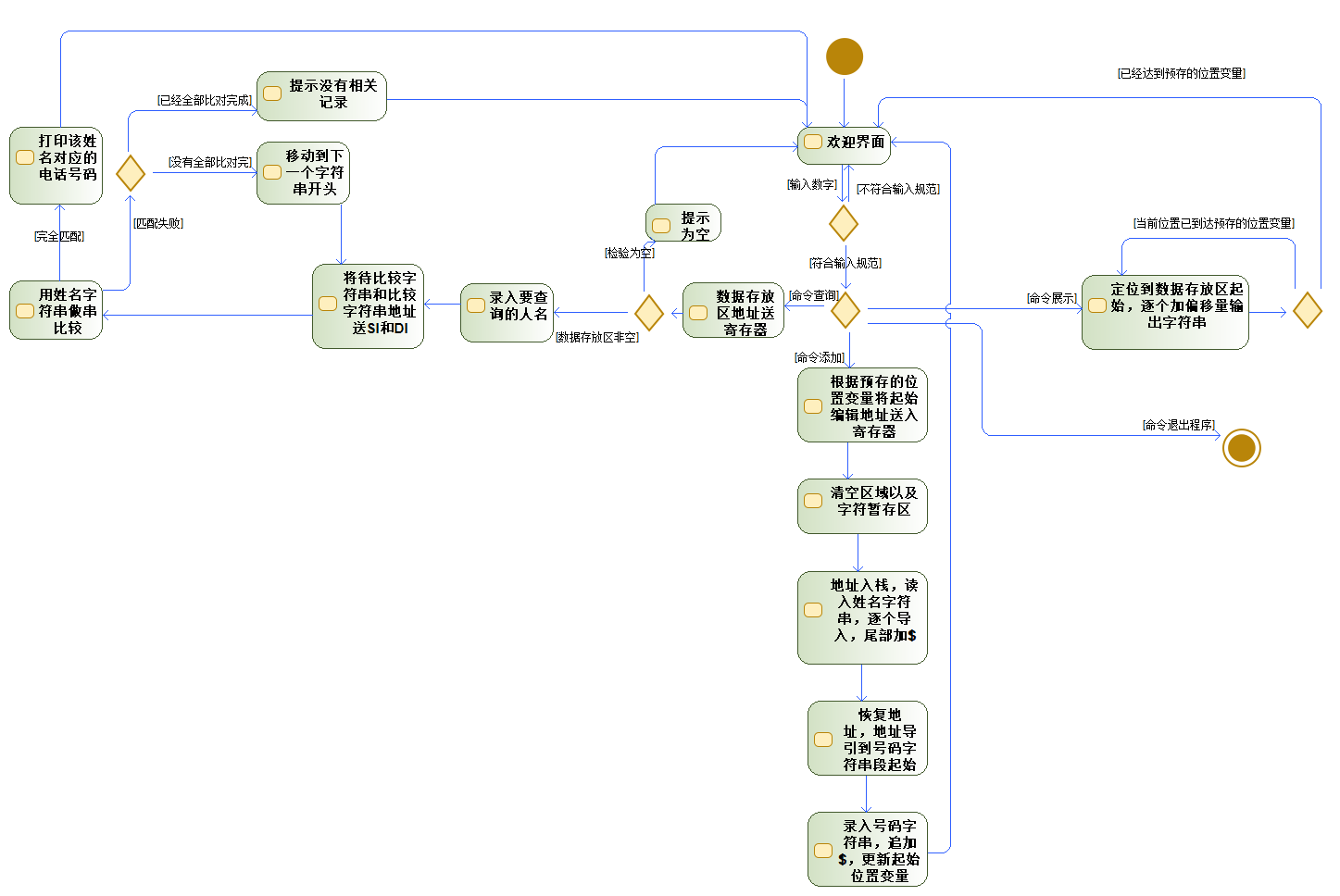
实验二：

（2）程序框图

实验一



实验二



（3）程序实现

实验一

mov ax, data

mov ds, ax

mov es, ax

mov dx, offset sentencepromp

mov ah,9

int 21h;打印提示输入句子

call reline;换行

mov dx,offset buf

mov ah,0ah

int 21h;录入句子

mov bx,0

mov bl,buf[1]

mov len,bl;送长度

call reline

mov dx,offset keywordpromp

mov ah,9

int 21h;提示输入关键字

call reline

mov dx,offset keyword

mov ah,0ah

int 21h;录入关键字

mov bx,0

mov bl,keyword[1]

mov keylen,bl;送长度

call reline

mov al, len

sub al, keylen

mov endindex,al;计算终结处

mov si,offset buf

mov di,offset keyword

add si,2

add di,2

call adjustsi

mov cx,0

mov cl,keylen

compare:

cmpsb

jne midfail

loop compare

call success

midfail:

mov al,shift

cmp al,endindex;如果已经到达边界，就不遍历下去了

jl prepare

jmp finalfail

prepare:

mov si,offset buf

mov di,offset keyword

inc shift

add si,2

add di,2

call adjustsi

mov cx,0

mov cl,keylen

jmp compare

finalfail:

lea dx,nomatch

mov ah,9

int 21h

call terminate

success:

lea dx,match

mov ah,9

int 21h

call terminate

ret

adjustsi:

mov cx,0

mov cl,shift

cmp cx,0

jle failsafe

call myadjustai

ret

myadjustai:

inc si

loop myadjustai

ret

failsafe:

mov cx,0

mov cl,keylen

jmp compare

terminate:

lea dx, pkey

mov ah, 9

int 21h

mov ah, 1

int 21h

mov ax, 4c00h ; exit to operating system.

int 21h

reline:

push dx

push ax

mov ah,02h

mov dl,0dh

int 21h

mov ah,02h

mov dl,0ah

int 21h

pop ax

pop dx

ret

实验二

data segment

welcome db "1 for adding,2 for consulting,3 for displaying.4 for exit.$"

order db 0

overflowprompt db "It is over the limit.Returning to title....$"

displayover db "Display is over.Returning to title$"

nonedisplay db "No item here,Please add some.Returning to title...$"

pkey db "press any key to exit...$"

initialwrong1 db "I cannot understand.Please give a correct order$"

addnameprompt db "input a name to save,below 20 words$"

addnumberprompt db "input a number under 8 words$"

consultnameprompt db "input a name to consult: $"

consultnumberprompt db "input a number to consult$"

totitlebuf db "Returning to title...$"

consultfailureprompt db "No result found.Return to menu...$"

consultsuccessprompt db "Record found,Number is:$"

ifempty db 1

pool db 300 dup(0)

savednum db 0

addnamebuf db 21,0,21 dup(0)

addnumberbuf db 9,0,9 dup(0)

deletenamebuf db 21,0,21 dup(0)

consultnamebuf db 21,0,21 dup(0)

currentindex dw -30;这个变量告诉我们，最新录入的一个的起始地址

ends

stack segment

dw 128 dup(0)

ends

code segment

start:

; set segment registers:

mov ax, data

mov ds, ax

mov es, ax

MOV bp,currentindex;30是一个基本的单元格长度，池子可容纳50个

; add your code here

call welcomer

call reline

cmp order,1

je adder

cmp order,2

je consult

cmp order,3

je display

cmp order,4

je terminate

mov dx,offset initialwrong1

mov ah,9

int 21h

call reline

jmp start

;;

welcomer:

push dx

push ax

lea dx, welcome

mov ah, 9

int 21h

call reline

mov ax,0

mov ah,1

int 21h

sub al,48;输入的是ASCII的字符

mov order,al

pop ax

pop dx ; output string at ds:dx

ret

;;

adder:

call reline

call adder\_numcheck

add bp,30

mov si,bp;作为基准的地址数值。

;cleaner:mov byte ptr pool[si],0

; inc si

; cmp [si],0

; jne cleaner

mov si,bp;清理完空间后回到原先的SI

mov currentindex,bp

mov dx,offset addnameprompt

mov ah,9

int 21h

call reline

mov dx,offset addnamebuf

mov ah,0ah

int 21h

mov bx,-1

s1:inc bx

cmp byte ptr addnamebuf[bx+2],0dh

jne s1;定位到键入信息的回车键之后

fillzero:mov byte ptr addnamebuf[bx+2],0

inc bx

cmp bx,20

jne fillzero;用0补完空位,免得本次的数据写入不能完全覆盖上次的。

mov byte ptr addnamebuf[bx+2],0dh

mov cx,20

mov bx,-1

store:inc bx

mov dl,byte ptr addnamebuf[bx+2]

mov byte ptr pool[si+bx],dl

loop store

mov byte ptr pool[si+bx+1],'$';封盖，方便输出

call reline

mov dx,offset addnumberprompt

mov ah,9

int 21h

call reline

mov dx,offset addnumberbuf

mov ah,0ah

int 21h

mov bx,-1

clear2:inc bx

cmp byte ptr addnumberbuf[bx+2],0dh

jne clear2

mov byte ptr addnumberbuf[bx+2],'$'

add si,21

mov cx,8

mov bx,-1

storenumber:

inc bx

mov dl,byte ptr addnumberbuf[bx+2]

mov byte ptr pool[si+bx],dl

loop storenumber

mov byte ptr pool[si+bx+1],'$'

call reline

mov dx,offset totitlebuf

mov ah,9

int 21h

call reline

inc savednum

mov ifempty,0;表示非空

jmp start

;;;;;;;

adder\_numcheck:

cmp savednum,10

jge start\_overflow

ret

start\_overflow:

push ax

push dx

mov dx , offset overflowprompt

mov ah,9

int 21h

pop dx

pop ax

jmp start

;;;;;;;

;;;;;;;

;;;;;;;

consult:

mov dx,offset consultnameprompt

mov ah,9

int 21h

mov dx,offset consultnamebuf

mov ah,10

int 21h

mov bx,-1

c1:inc bx

cmp byte ptr consultnamebuf[bx+2],0dh

jne c1

c2:mov byte ptr consultnamebuf[bx+2],0

inc bx

cmp bx,20

jne c2

mov byte ptr consultnamebuf[bx+2],0dh

call reline

mov bp,offset pool

sub bp,30

push currentindex

push bx;

lea bx,pool

add bx,currentindex

mov currentindex,bx

pop bx;

outer:

mov cx,20

add bp,30

;;;;?

cmp bp,currentindex

jg fail

MOV SI,OFFSET consultnamebuf

add si,2

MOV DI,bp

jle inner

inner:

cmpsb

jnz outer

loop inner

jmp success

fail:

mov dx,offset consultfailureprompt

mov ah,9

int 21h

call reline

pop currentindex

jmp start

success:

mov dx,offset consultsuccessprompt

mov ah,9

int 21h

MOV dx,bp

add dx,21

mov ah,9

int 21h

pop currentindex

call reline

jmp start

jmp start

;;;;;;;

display:

call reline

cmp ifempty,1

je emptyexc

mov bx,0

mov bp,offset pool

sub bp,30

circle:

push bp

add bp,30

sub bp,offset pool

cmp bp,currentindex

jg jumpout

pop bp

add bp,30

mov dx,bp

mov ah,9

int 21h

call reline

add dx,21

mov ah,9

int 21h

call reline

jmp circle

;circle

emptyexc:

push dx

push ax

mov dx,offset nonedisplay;无记录的错误提示

mov ah,9

int 21h

call reline

pop ax

pop dx

jmp start

jumpout:

mov dx,offset displayover

mov ah,9

int 21h

call reline

jmp start

jmp start

;;;;;;;

reline:

push dx

push ax

mov ah,02h

mov dl,0dh

int 21h

mov ah,02h

mov dl,0ah

int 21h

pop ax

pop dx

ret

terminate:

lea dx, pkey

mov ah, 9

int 21h ; output string at ds:dx

; wait for any key....

mov ah, 1

int 21h

mov ax, 4c00h ; exit to operating system.

int 21h

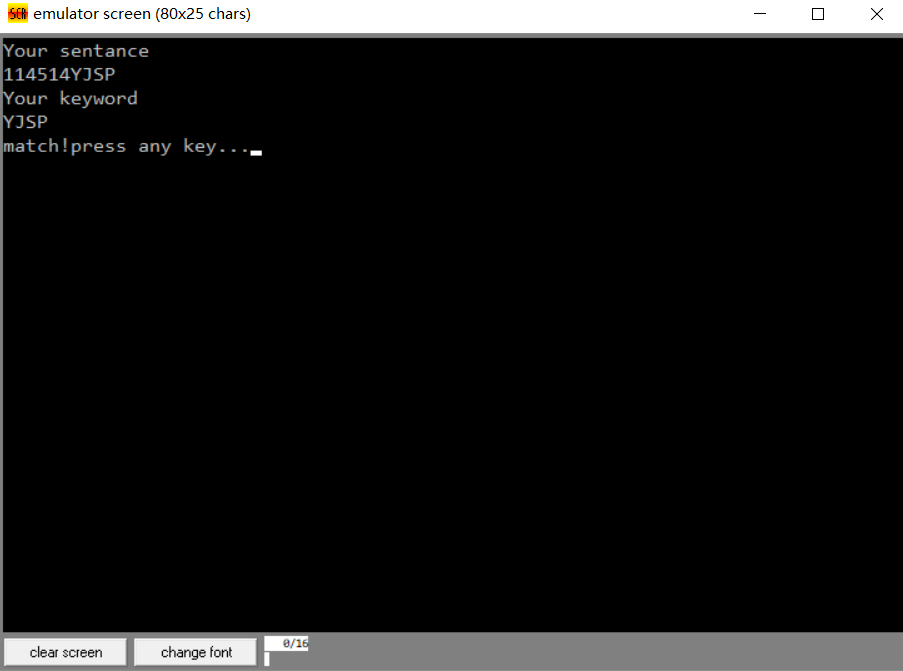
ret

ends

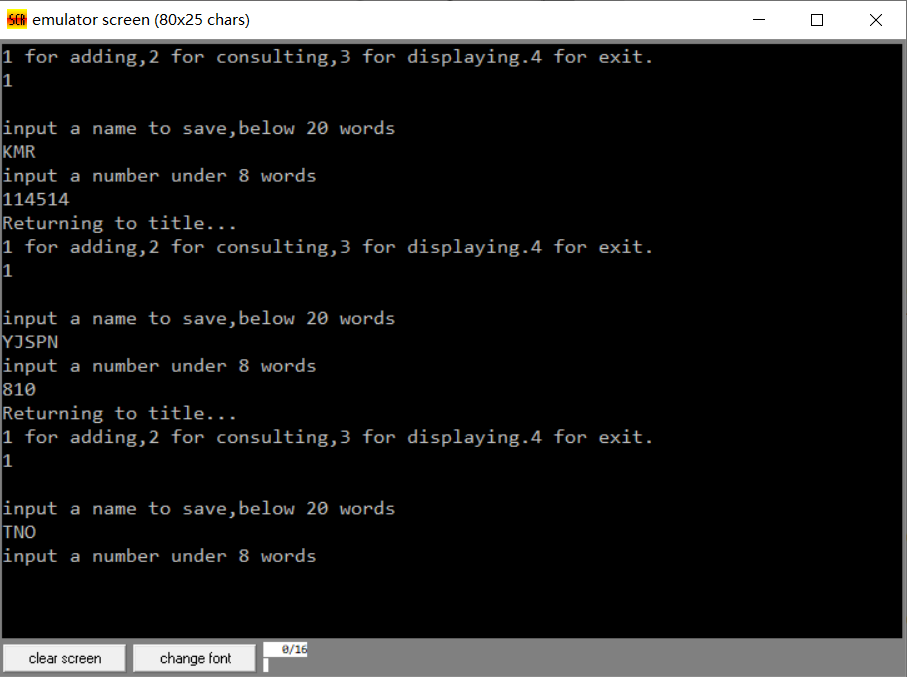
end start ; set entry point and stop the assembler.

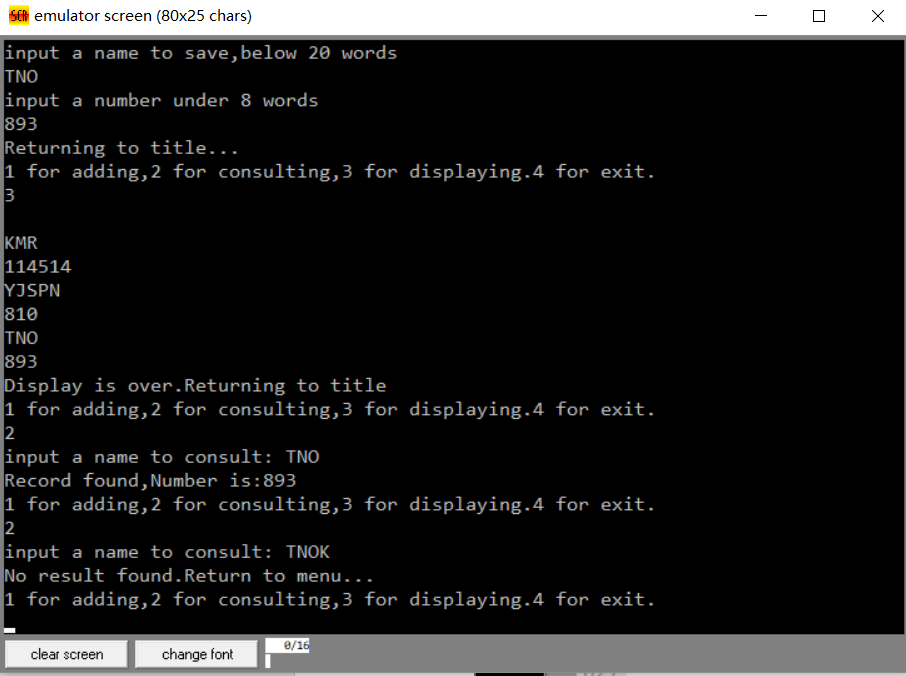
（4）实验结果

实验一



实验二





**三、课程设计（综合实验）总结或结论**

四号实验可谓是所有实验题目里最难的了，如果是使用C++,JAVA这种高级语言，这种功能我只需一个小时就能做出来，而汇编实现却零零散散花了我五天，这个大工程涉及到大量的条件判断，所以源代码里面有各式各样的跳转指令，实验四的一个核心就是串操作，第二部分的实验更是涉及到了如何在连续区域里定位比对字符串，通过这次试验，我极大加深了对于汇编的认知，并且深刻体会到使用高级语言的幸福感来之不易，忆苦思甜，今后必须更加注重程序编写的结构合理，构思合理，算法优化，争取早日为我国软件工业发展献出自己的力量

**四、参考文献**

《IBM-PC汇编语言程序设计（沉美明第二版）》