

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：查找匹配字符串与电话号码本

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

**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**

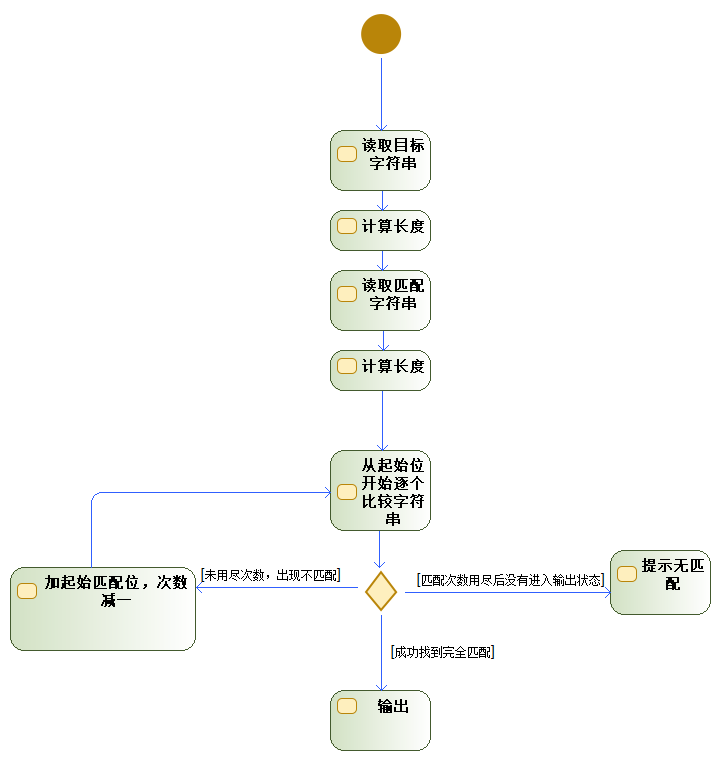
（1）设计思路

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。

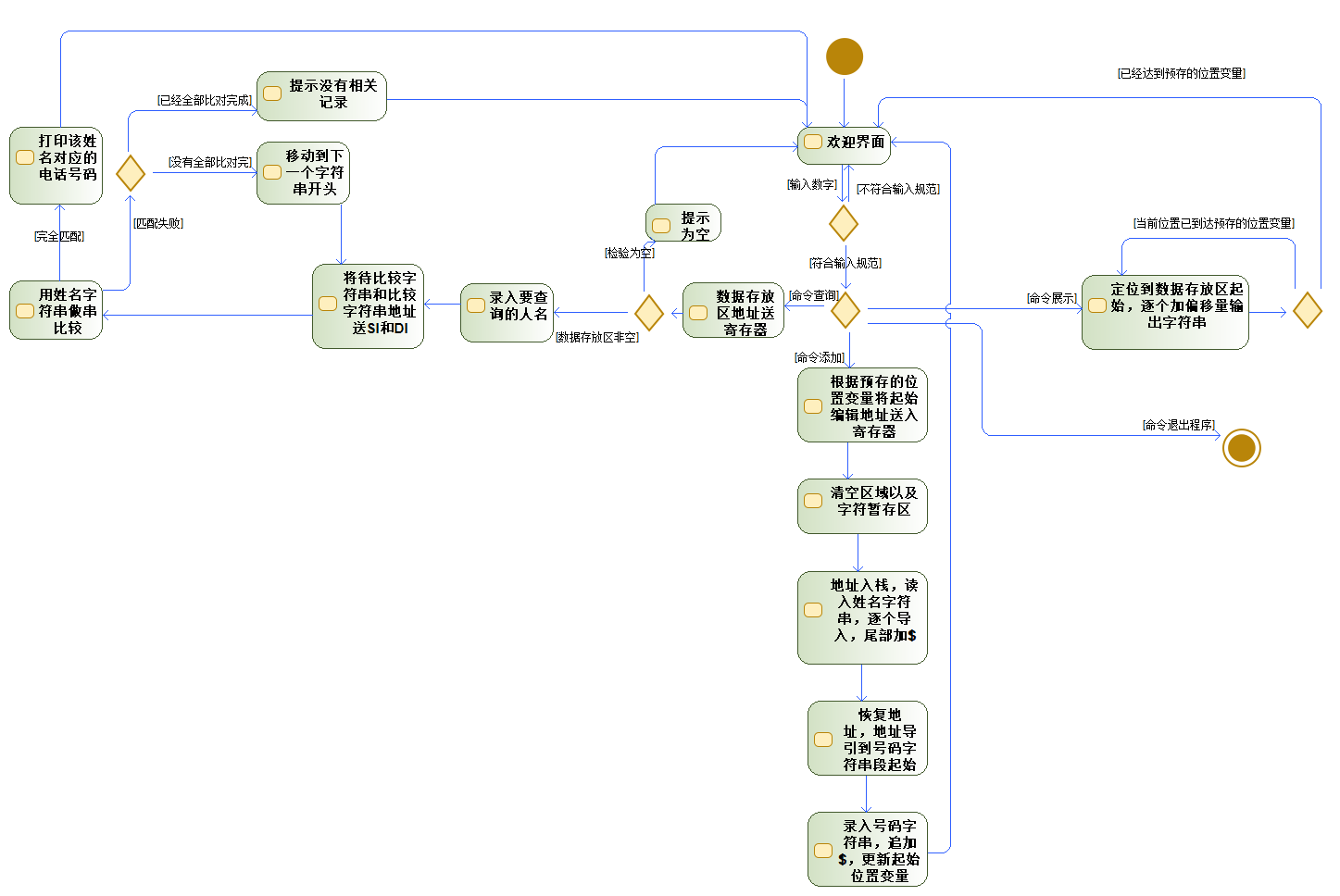
实验二：

（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

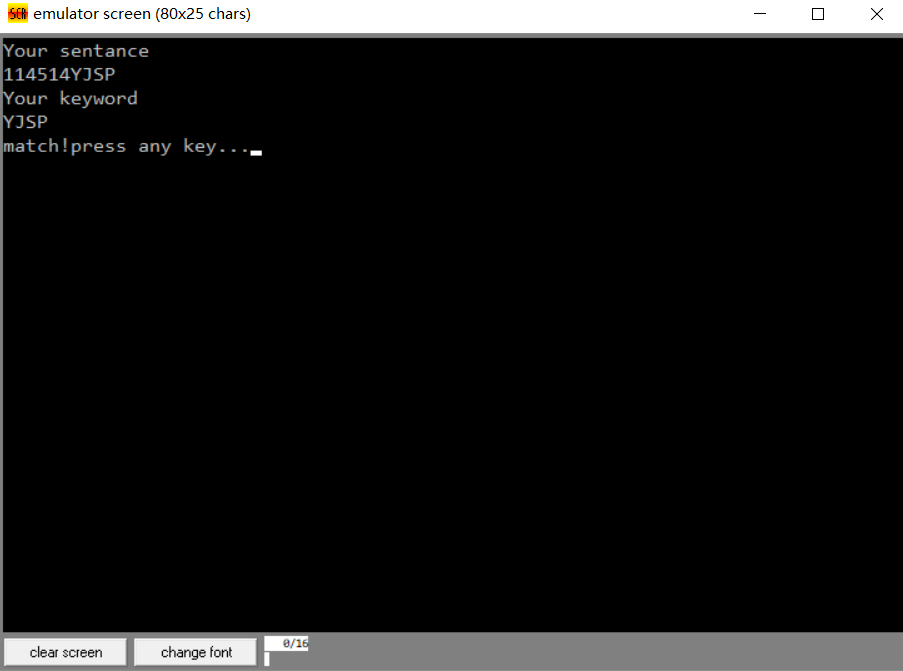
ret

ends

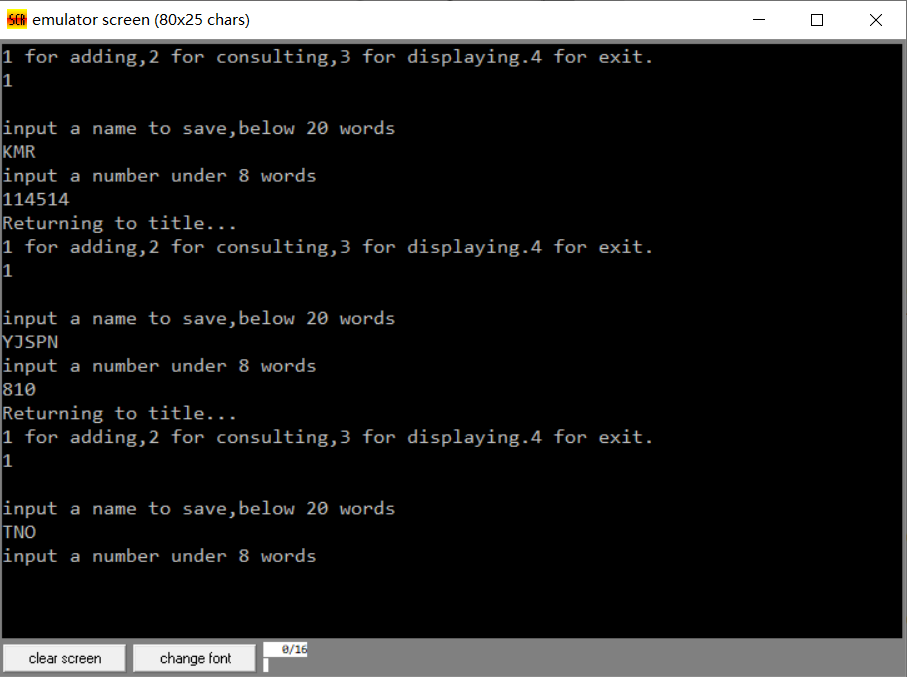
end start ; set entry point and stop the assembler.

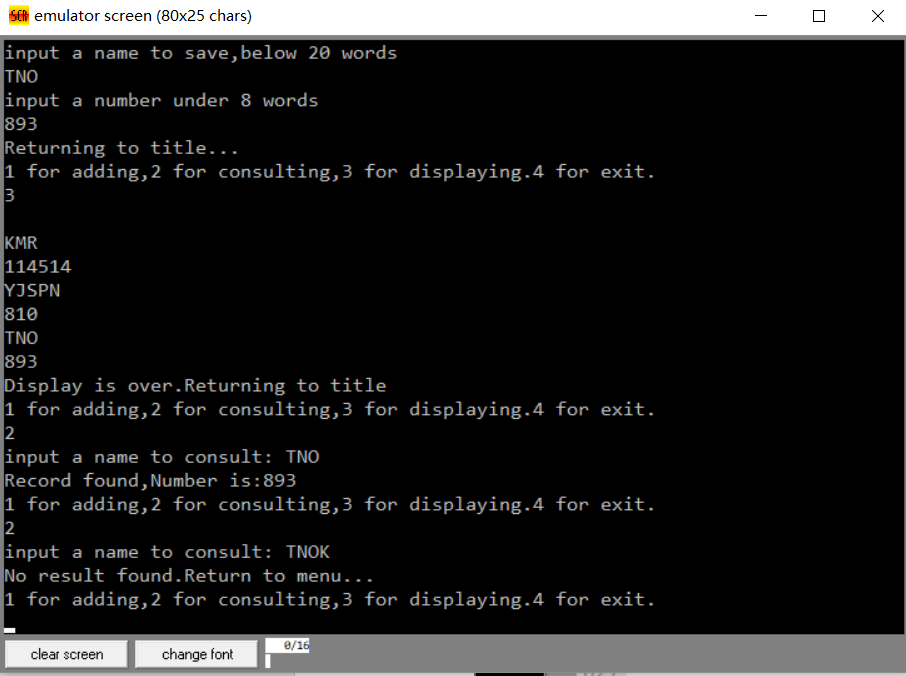
（4）实验结果

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）》