X86 汇编程序设计第三次实验作业

手写代码见最后

- 1. 编写一道完整汇编程序,实现以下要求(提示:参考讲义例题修改):
 - (1) 编写子程序 Memmove, 将处于同一数据段的串 String1 复制到 String2。 子程序入口参数为: DS:SI 指向 String1, ES:DI 指向 String2。
 - (2) 数据段先定义 String1 为你的姓名的汉语拼音,在此串前后各留一点空余串。构造 String2 的首地址分别为(1)与 String1 不重叠; (2) String2 地址在 String1 前,但有重叠; (3) String2 地址在 String1 后,但有重叠。
 - (3) 主程序分三次调用 Memmove,每次调用前显示 String1,调用后显示 String1,String2。

```
::\MASM\BIN>masm 3-1.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.
Object filename [3-1.OBJ]:
Source listing [NUL.LST]: 3-1
Cross-reference [NUL.CRF]: 3-1
 50610 + 465934 Bytes symbol space free
     0 Warning Errors
     0 Severe Errors
C:\MASM\BIN>link 3-1.obj
Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.
Run File [3-1.EXE]:
List File [NUL.MAP]: 3-1
Libraries [.LIB]:
C:\MASM\BIN>
C:\MASM\BIN>3-1.exe
zhaoliangxuan zhaoliangxuan zhaoliangxuan
zhaoliangxuan ngxuan zhaoliangxuan
ngxuan ngxuan ngxuan
C:\MASM\BIN>
```

- (1) 常规的 strcmp
- (2) String2 在 String1 前,故 String2 的尾覆盖了 String1 的头
- (3) 在(2) 的基础上, String2 地址在 String1 后, 只复制了"残存"的 String1

2. 编写一道完整汇编程序,实现:从键盘输入一个字符串 String2 (测试时输入你的姓名的汉语拼音),与内存中的已有字符串 String1 比较。根据比较结果,显示"XXXX>=<YYYY"。(提示:输入、比较、显示编写为子程序或宏;显示结果时,分段显示 XXXX,字符=(或<,>),YYYY)。

```
C:\>cd masm\bin
C:\Masm\BIN>masm 3-2.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

Object filename [3-2.0BJ]:
Source listing INUL.LST]: 3-2
Cross-reference INUL.CRF]: 3-2

50534 + 466010 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\Masm\BIN\link 3-2.obj

Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

Run File [3-2.EXE]: 3-2
List File INUL.MAP]: 3-2
Libraries [.LIB]:
C:\Masm\BIN>_
C:\Masm\BIN>_-
C:\Masm\BIN>_-2.exe
```

已有字符串 String1 为 liangxuanxuan,输入字符串为 zhaoliangxuan。

按照字典序,有 liangxuanxuan < zhaoliangxuan。

liangxuanxuan < zhaoliangxuan

C:\MASM\BIN>_

3. 编程实现 (讲义 149 页第 6 题):

6. 编写一个有主程序及子程序结构的程序模块。主程序从键盘接收一串字符建立串 STRING,然后将 STRING 的首地址、串长度作为参数通过堆栈传递给段内调用的子程序,同时在 AL 中放入要查找的字符。子程序 FIND 查找 AL 中的字符在 STRING 中出现的次数,并将出现次数放入 AX 返回。

```
C:\MASM\BIN\masm 3-3.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

Object filename [3-3.0BJ]:
Source listing [NUL.LST]: 3-3
Cross-reference [NUL.CRF]: 3-3

50572 + 465972 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\MASM\BIN\link 3-3.obj

Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

Run File [3-3.EXE]:
List File [NUL.MAP]: 3-3
Libraries [.LIB]:
C:\MASM\BIN\>
C:\MASM\BIN\>
C:\MASM\BIN\>
C:\MASM\BIN\>
3
C:\MASM\BIN\>
```

设置的要查找的字符为'a',输入的串 STRING 为'zhaoliangxuan',故输出 3

4. 选做题: 先构造一个不等长的单词表(区分大小写), 然后进行字典式排序, 再输入一个单词(你的姓名的汉语拼音), 插入到此单词表中。要求输出显示排序前后的单词表。(此题不用手写, 只需提交运行结果及源程序)。

```
C:\MASM\BIN\masm 3-4.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

Object filename [3-4.0BJ]:
Source listing [NUL.LST]: 3-4
Cross-reference [NUL.CRF]: 3-4

50496 + 449664 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\MASM\BIN\link 3-4.obj

Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

Run File [3-4.EXE]:
List File [NUL.MAP]: 3-4
Libraries [.LIB]:

C:\MASM\BIN\
```

```
beihang
SCSE
X86
2020
zuoye3
xuanzuoti
2020
SCSE
X86
be i hang
xuanzuoti
zuoye3
zhao l i angxuan
2020
SCSE
X86
be i hang
xuanzuoti
zhaol iangxuan
zuoye3
C:\MASM\BIN>
```

单词表为: beihang, SCSE, X86, 2020, zuoye3, xuanzuoti

排序后为: 2020, SCSE, X86, beihang, xuanzuoti, zuoye3

插入 zhaoliangxuan 再排序后为: 2020, SCSE, X86, beihang, xuanzuoti, zhaoliangxuan,

zuoye3

手写代码:

3-1:

3-1 MemMo	ove.			MOV	OX.OFFSET. STR1
STACK	SEGMENT	PARA, STACK		MOV	AH. 9H
STACK-AREA	DW	Iwoh DUP(?)		INT	عالم.
STACK - TOP	EQU	4-STACK_AREA		MOV	DX. OFFSET SPACE
>TACK	EMDS	1 STACKE MICH		MOV	AH.ojh
		8. 4.		INT	214
DATA	SEGMENT	PARA		MOV	Dx , Bx
STRI		chaoliargxuon', '\$'		MOV	AH. 94
STRZ	DB	30, DUP(?)		INT	114 NEW-LINE
SPACE	OB	жн. '4'		MOV	DX. FFET PARE
NEW-LINE	DB	ODH . OAH . '\$'		MOV	AH. of H
ATA	EMS			INT	214
CODE	SEGMENT	To de Taul	DISP-STR	EMP	
	ASSUME CS:1	LOPE, OS: DATA, SS: STACK		1200	
NIAM	PROC	FAR	MEM-MOVE	PROC	
	MOV	AX, STACK		LODSB	
	MOV	55. AX	MM-LOPPO	STOSB	
	MOV	SP, STACK-TOP		cmp. R	١٤. ١٤
	MOV	AX, DATA		JNZ 1	m-upp
	MOV	DS · AX		RET	•
	WOV	ES, AX	MEM-MOVE	ENOP	
	W ₀ V	BX. OFFIET STR2	CODE	EMD5	
	CALL	DISP.STR		EMD	MAIN
	MOV	BX. OFFSET STRI-7 DISY-STR			
	MOV	BX OFFSE STRITS			
	CALL	DJSP-STR			
	MOV	Ax.4coor			
	INT	414			
MAIN	EMPP				
	•				
DIG-STR	proc				
	MOV	DX, OFFSET STRI			
	WOV	A4. जीम			
	INT	عالم			
	MOV	DX, OFFSET SPACE			
	MOV 7.IT	A4, ज्म			
	INT	214			
	MOV	SI. OFFSET STRI			
	MOV	DI, BX			
	CAU	MEM- MOVE			
					\bigcirc
					U

3-2:

mm 7				
3-2 STACK	CTALAZ. AT		MoV	ay. Result
>TACK-AREA	SECMENT PARA. STACK		cnp	Ax.I
STACK_TOP	DW [NOH, DOPC?)		AL	MAIN-A
STACK	ERU \$- STACK-AREA		15	MIN-Z
1	ĒMX		MOV	DX, OFFSET BELOW
DATA	SEGMENT PARA		JWb	PRINT
RESULT	DW ?	MAIN_A:	WOV	DX. OFFSET ABOVE
STRI	DB 'zhaoliangxuan'.'\$'		JWb	PRINT
LEN	E80 171	MAJN_Z:	MOV	DX. OFFSET EQUAL
IN-BUFF	DB LEN-1	PRINT:	CALL	PIS-STR
	DB 7		MOV	DX. OFFSET IN-BOF+2
	DB LEN DUP(?)		CALL	DIS-STR
ABOVE	DB '>', '\$'		MOV	Ax.4cmH
BELOW	DB '<'.'\$'		INT	411
EQUAL	DB '='.'\$'	MAIN	EMPP	
SPACE	DB 20H. '4'	- T. A. I.		
NEW-LIME	DB ODH, JAH. '\$'	IMPUT-STR:	PROC	A
DATA	EMDS		WOV	AH. OAH
CODE	SEGMENT		MOV	DX, OFFSET IN-BUF
	ASSUME 'S: COIDE. DS: DATA, SS: STACK	-	INT	214
MIAM	Proc FAR		WOV	CL, IN-BUF+1
	mov ax, stack		MOV MOV	CH. CH SI. OFFSET IN-BUF +2
	MOU SS. AX		ADD	S1. CX
	MOV SP. STACK_TOP		MOV	BYTE PTR ESII. '\$'
	MOV AX. DATA	INPUT_STR	EMP	1
	MOV DS. AX			
	MOV ES, AX.	STRCMP	PROC	
	CALL JAPOT-STR		PUSH	CX
	MOV SI OFFSET IN-BUE+2		PUSH	ВX
	MOU CL, IN-PUT-11		PUSH	Ax
	*CH-CH	LP3:	CMP	BYTE PTR [SJ]. '\$'
	pusu SI-	1.	75	LP3-1
	PUSH C*			
	MOV AL, 'a'		cmp	BYTE PTR [D]].'4'
			75	A
	cau input. STR		JMP	493-2
	MOV 31. OFFSET STRI	LP3-1:	cmp	BYTE. PTRIDII.'S'
	MOV DI. OFFET IN. BUF+2		15	2
	CALL STREMP		JWb	В
	MOV DX, OFFSET. STRI			
	CALL DIS. STR			②

```
43-2:
                     AL, BYTE. PTR[5]].
            MOV
            cmp
                     AL . BYTE PTR [D]]
            JA
                      B
            JB
                      5]
            INC
                     DI
            INC
            JMP
                     43
                      RESULT. 0
            MOV
Bi
            JMP
                      RETURN
             MOV
                      RESULT, 2
A:
            JMP
                      RETURN
                      RESULT.1
₹:
             MOV
RETURN.
             Pop
                       AX
             POP
                       BX
             POP
                       CX
             RET
STREMP
             EMP
DIS-STR
              PROC
                      A4.094
             WOV
              INT
                      214
             RET
DIS-STR
             EMPP
              EMDS
CODE
             EMD
                      MAIN
```

3-3:

3-3		,		MOV	Ax, 4 cupy
STACK	SEGMENT	PARA. STACK		IMI	111
STACK AREA	DW	(wh. Dup(?)	MIAM	EMOP	
STACK - TOP	EQU	4-STACE-APEA	FIND	Dear	
STACK	EMS		FIND	proc	BP. SP
DATA	SEGMENT	para		MOV	CX-[B]+2]
FIEL	PB	zhootingran -		WOV	SI. [BP+4]
57K2	DB-	ho . DUPC ?>		PUSH	DX
LEN	EQU	12)		PUSH	ВХ
IN-BUFF	DB	LEN-1			
	DB	?		XOR	BX . BX
	DB	LEN. DUP(?)		WOV	DL. AL
NEW-LINE	PB	ODH.OAH.'}		CLD	
DATA	ENDS		41:	LODSB	111
	SEGMENT			cmp	AL. '\$'
CODE		s:codē,ds:data.ss:stack		J₹	41-1
	Proc	FAR	r.	cmp	AL.DL
MIAM	MOV	AX, STACE		JNZ	491
	MOU	ss. AX		INC	ВX
	WOV	SP. STACK_TOP		JWb	LPI
	MOV	AX DATA	41-1	WoA	AX . BX
	VON	DS, AX		POP	Вх
	MoV	ES, AX.		POP	DX
	CALL	IMPUT_STR		RET	
	WOV	SI. OFFSET IN-BUF+2	FINDO	ENDP	
	WOY	CL, IN-BUF+1			
	XOR	CH. CH	IMPUT_STR.	PROC	
	PUSH	SI		MOV	AH, OAH
	PUSH	CX		WOV	dx. offset IN-BUF
	MOV	AL.'a'		INT	214
	CALL	ED PLM)		WOL	CL. IN-BUFTI
	PUSH	Ax		YOR	CH. CH
	MOV	DX, OFFSET NEW-LINE		WOV	SI.OFFSET IN_BUF+2 SI.CX
	Vav	AX.09H		ADD	
	INI	٦١٢		MOV PET	भेराह प्रारह्या, 'ई'
	POP	XA	TADIT CTO	RET	
	MOU	DX . AX	IMPUT_STR	EMPP	
	OR	DL, 304	CODE	ENDS	
	WOA	AH. 2		END	MAIN
	INT	214			