山东大学 计算机科学与技术 学院

汇编语言 课程实验报告

学号：202120130276	姓名：王云强	班级：21.2 班
实验题目：实验 2.4		
实验学时：2	实验日期：2023.12.19	

实验目的：

1、巩固子程序设计中所涉及的知识点。

2、学会在自编程序中利用子程序设计的理论与技巧。

3、学会在自编程序中使用系统调用和简单的表格数据结构。

实验环境：Windows10、DOSBox-0.74、Masm64

源程序清单：

1. EX11.ASM（实验 2.4 源程序）

编译及运行结果：

编译结果：

```
C:\>masm EX11
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [EX11.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  51640 + 448520 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link EX11

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

Run File [EX11.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment
```

运行结果：

```
C:\>EX11
3
Input name:abc
Input a telephone number:12345678
Input name:bca
Input a telephone number:23456789
Input name:cab
Input a telephone number:34567890
Do you want a telephone number?(Y/N)Y
name?cab
name                tel.
cab                 34567890
Do you want a telephone number?(Y/N)Y
name?abc
name                tel.
abc                 12345678
Do you want a telephone number?(Y/N)N
abc                 12345678
bca                 23456789
cab                 34567890
```

问题及收获：

一、如何存储所有的姓名以及对应的电话？

在本程序中，我们用一个表格（相当于一个二维数组）来存储

```
THE_NUM         DB              20 DUP(28 DUP(' '))                 ;定义二维数组记录姓名以及对应电话
```

该操作相当于提前在内存中开出一个具有 20 个大单元地址内容，每个大单元地址内容中储存这 28 个字节的位置，都存放着空格。这样相当于开出了 20×28 的地址单元来储存姓名和对应的电话号。

二、实验中涉及到大量的输入，如何进行输入？

参考以往的实验，首先考虑到的是输入数据的数量，这里我们根据题目要求多次调用 MOV AH, 01H，INT 21H。考虑到数据范围，所以我们至少调用两次（输入最少是个个位数，一次用来读数据，一次用来读回车）。然后对输入的字符进行处理转换，即可得到相应的数字。

```
                MOV             AH, 01H                         ;根据题目要求多次调用
                INT             21H
                SUB             AL, 30H
                MOV             COUNT,AL

                MOV             AH, 01H
                INT             21H
                CMP             AL, 13
                JE              OVER

                SUB             AL, 30H
                ADD             COUNT, AL
                ADD             COUNT, 9

                MOV             AX, 0
                MOV             AH, 01H
                INT             21H
                CMP             AL, 13
                JE              OVER
OVER:
                MOV             CL, COUNT
                MOV             CH, 0
                LEA             DI, THE_NUM
```

之后我们还是像之前输入字符串一样，先定义相关的缓冲区：

```
NAMING          LABEL           BYTE                            ;定义字符串记录姓名
N_MAX_LEN       DB              21
N_LEN           DB              ?
THE_NAME        DB              21 DUP(' ')

PHONING         LABEL           BYTE                            ;定义字符串记录电话
P_MAX_LEN       DB              9
P_LEN           DB              ?
THE_PHONE       DB              9 DUP(' ')
```

接下来就是输入，我们使用 LOOP 指令，循环输入，但是需要注意的是，每次我们都需要将缓冲区清空才能继续输入，不然上一次输入的数据会和这一次输入的数据存在一起，从而导致错误。在输入完数据以后，我们还应该将其存放到对应的二维数组中（即上面我们定义的那个）。

```
INPUTING:
                CALL            CLEAR                           ;这是为了先把原来存在的数据清空
                CALL            INPUT                           ;这是输入部分
                CALL            STORE
                LOOP            INPUTING
```

CLEAR 子程序：该子程序的作用是为了把原来存在缓冲区的数据先清空，实现代码如下：

```
;-------------------------------------------------------------------------
CLEAR           PROC            NEAR                    ;每次清空输入时候的两个字符串
                PUSH            CX
                PUSH            AX
                PUSH            DI
                LEA             DI, NAMING
                MOV             AL, 32
                MOV             CX, 20
                REP             STOSB
                LEA             DI, PHONING
                MOV             AL, 32
                MOV             CX, 9
                REP             STOSB
                POP             DI
                POP             AX
                POP             CX
                RET
CLEAR           ENDP
```

INPUT 子程序：该子程序的作用是为了本实验的输入过程，实现的代码如下：

```
INPUT           PROC            NEAR
                PUSH            CX
                PUSH            SI
                LEA             DX, MESSAGE1
                MOV             AH, 09H
                INT             21H
                LEA             DX, NAMING
                MOV             AH, 0AH
                INT             21H

                MOV             CH, 0
                MOV             CL, N_LEN
                LEA             SI, NAMING              ;输入姓名
                ADD             SI, CX
                INC             SI
                INC             SI
                MOV             BYTE PTR[SI], 32

                CALL            CRLF
                LEA             DX, MESSAGE2
                MOV             AH, 09H
                INT             21H
                LEA             DX, PHONING
                MOV             AH, 0AH
                INT             21H

                MOV             CH, 0                   ;输入电话
                MOV             CL, P_LEN
                LEA             SI, PHONING
                ADD             SI, CX
                INC             SI
                INC             SI
                MOV             BYTE PTR[SI], 32
```

STORE 子程序：该子程序的作用是为了把输入得到的数据存放到相对应的二维数组中，实现代码如下：

```
STORE           PROC            NEAR
                PUSH            CX
                LEA             SI, THE_NAME
                MOV             CX, 20
                REP             MOVSB
                LEA             SI, THE_PHONE
                MOV             CX, 8
                REP             MOVSB
                POP             CX
                RET
STORE           ENDP
```

之后就是查询部分的输入，此处我们先读入单个字符判断是 Y 还是 N，然后在决定是否需要输入姓名，姓名的输入方式跟上面姓名的输入方式是一样的，不再重述。

三、为什么调用 REP MOVSB（或 MOVSW）后，没有正确地将 SI 的内容复制到 DI 对应地址单元？

这是因为 REP MOVSB 指令是将 DS:SI 中的 CX 个内容复制到 ES:DI 对应的地址单元开始的 CX 个内容。所以如果要使用这个指令，需要让 ES 在开头也与 DS 一样指向数据段。

四、当输入完数据以后，该如何进行查询？

在实验中，由于我们使用一个数组来存数据，所以我们就需要到数组中找到相应的姓名然后输出。具体为先输入查询姓名，然后先输出换行，之后查找电话号码，并将查询到的电话号码存在放 THE_TMP（THE_TMP 也是一个字符串）中，所以我们每次还需要清空一下 THE_TMP，也是为了防止跟上一次的数据重合。

该部分代码如下：

```
FINDING:
        CALL    CRLF
        LEA     DX,MESSAGE4
        MOV     AH, 09H
        INT     21H

        LEA     DX, NAMING                      ;输入姓名
        MOV     AH, 0AH
        INT     21H
        MOV     CH, 0
        MOV     CL, N_LEN
        LEA     SI, NAMING
        ADD     SI, CX
        INC     SI
        INC     SI
        MOV     BYTE PTR[SI], 32

        CALL    CRLF
        CALL    FIND                            ;在二维数组里面进行查找，找到了就输出信息即可
        CALL    CLEAR1
        LEA     DX, MESSAGE5
        MOV     AH, 09H
        INT     21H
        CALL    CRLF
        LEA     DX, THE_TMP
        MOV     AH, 09H
        INT     21H
        CALL    CRLF
        JMP     ASKING
EXITING:
        MOV     AH,4CH
        INT     21H
MAIN    ENDP
```

其中子函数 CRLF 是输出回车换行，查找部分主要在 FIND 子函数。下面主要解释下该函数，该函数先是将输入的名字存放到 ASK_PHONE 中（ASK_PHONE 是一个字符串），然后再去对比二维数组中存放的名字，如果相同的话就取出电话号码，使用 REPZ CMPSB 指令来进行比较，并将取出的电话号码放在 THE_TMP 中，相关代码如下：

```
FIND    PROC    NEAR
        PUSH    CX

        MOV     CL, NAMING + 1
        MOV     CH, 0
        LEA     SI, NAMING
        INC     SI
        INC     SI
        LEA     DI, ASK_PHONE
        REP     MOVSB

        LEA     SI, ASK_PHONE
        LEA     DI, THE_NUM
        MOV     BX, DI
        MOV     CL, COUNT
        MOV     CH, 0
I3:
        PUSH    CX
```

```
I3:
                PUSH            CX
                MOV             CH, 0
                MOV             CL, 20
                REPZ            CMPSB
                JE              FIND_EXIT
                POP             CX
                ADD             BX, 28
                MOV             DI, BX
                LEA             SI, ASK_PHONE
                LOOP            I3
                JMP             TO_EXIT
FIND_EXIT:
                POP             CX
                MOV             SI, BX
                LEA             DI, THE_TMP
                MOV             CX, 14
                CLD
                REP             MOVSW
TO_EXIT:
                POP             CX
                RET
FIND            ENDP
```

五、如何输出排序后的名字和电话号码？

要实现排序以后的输出，这里我们采用冒泡排序，即我们用汇编语言的形式实现一个冒泡排序来完成，相关代码如下：

采用双重循环，两两之间进行比较，如果满足条件就进行交换

```
;-------------------------------------------------------------------------
SORT            PROC            NEAR
                MOV             CH, 0                           ;CH = i
OUTER_FOR:                                                      ;for(int i=1;i<n;++i)
                INC             CH
                CMP             CH, COUNT
                JNB             BREAKING
                LEA             SI, THE_NUM
                LEA             DI, THE_NUM + 28                ;DI = a[j+1], SI = a[j]
                MOV             CL, 0                           ;CL = j
INNER_FOR:
                MOV             BL, COUNT
                SUB             BL, CH
                CMP             CL, BL                          ;j < COUNT - i - 1
                JNB             OUTER_FOR                       ;for(int j=0;j<COUNT - i - 1;++j)
                PUSH            CX
                MOV             CX, 20
                MOV             BX, DI
                MOV             AX, SI
                REPE            CMPSB
                JBE             NEXT_FOR                        ;a[j] > a[j + 1]----->CHANGE
                CALL            CHANGE                          ;a[j] <= a[j+1]------>next i
NEXT_FOR:
                POP             CX
                MOV             DI, BX
                MOV             SI, AX
                ADD             SI, 28
                ADD             DI ,28
                INC             CL
                JMP             INNER_FOR
BREAKING:
                RET
SORT            ENDP
```

```
;-------------------------------------------------------------------------
CHANGE          PROC            NEAR
                MOV             SI, BX                          ;temp = a[j+1]
                LEA             DI, THE_SORT
                MOV             CX, 14
                REP             MOVSW

                MOV             DI, BX                          ;a[j+1] = a[j]
                MOV             SI, AX
                MOV             CX, 14
                REP             MOVSW

                MOV             DI, AX                          ;a[j] = temp
                LEA             SI, THE_SORT
                MOV             CX, 14
                REP             MOVSW
                RET
CHANGE          ENDP
;-------------------------------------------------------------------------
```

六、为什么有时候没法正确查询，且最后输出的时候会多输出一些内容（如 输入为 a, b ,c 时，在查询完 a 后，查询 b 时有误）

这是因为程序是利用一个固定的表项接受输入，如姓名固定输入在 NAMING 中，电话号码固定输入在 PHONING 中，而多次输入之间没有进行清空，导致前面假设输入了 a 在姓名中，后面在输入 b 的时候， 实际读入的是 ba，导致出现了问题。解决方法就是在每次输入之前或者之后，

将输入表项进行清空，使得多次输入之间不会互相影响。

收获：

1.这次程序重新复习、巩固了一下关于表格数据的使用方法，掌握了汇编语言二维数组的使用方法（本质同一维数组一致）。

2.巩固了关于字符串指令的使用方法，如 MOVSB 和 CMPSB，对其使用有了更清楚地认识，同时对子程序的调用有了更深的理解和认识。

3.关于汇编语言的 DEBUG 能力得到了提升，懂得更灵活的使用 DEBUG，同时对于细节的把握更深刻了。

4.对于汇编语言与高级程序语言相互转换有了更深的理解，如在本实验中使用汇编语言模拟了 C 语言中的冒泡排序，无论是循环过程还是分支过程都有了更灵活、深刻的把握。