

高级汇编语言程序设计

学 院：_____

专 业：_____

姓 名：_____

学 号：_____

指导教师：_____

一、实验目的

1. 掌握高级汇编语言技术的方法。
2. 掌握宏定义、宏调用方法。
3. 熟悉宏定义中的参数形式。

二、实验准备

1. 复习教材中有关宏汇编的内容。
2. 编写一条宏指令 CLRB，完成用空格符将一字符区中的字符取代工作。字符区首地址及其长度为变元。

3. 请编写宏指令宏：BIN_SUB 完成多个字节数据连减的功能：

RESULT ← (A-B-C-D-…)

要相减的字节数据顺序存放在首地址为 OPERAND 的数据区中，减数的个数存放在 COUNT 单元中，最后结果存入 RESULT 单元。

三、实验内容

调试并调用编写的宏指令。

- 1、编写一条宏指令 CLRB，完成用空格符将一字符区中的字符取代工作。字符区首地址及其长度为变元。

代码如下：

宏指令：

```
CLRB    MACRO    N, STRING
        MOV     DI, OFFSET STRING
        MOV     CX, N
        MOV     AL, 32          ;;32 是空格的 ASCII 码
```

TIHUAN:

```
        CMP     CX, 0
        JZ      EXIT
        MOV     [STRING], AL
        DEC     CX
        INC     DI
```

JMP TIHUAN ;;从 DI 所指的内存开始,将连续的 CX 个字节写成 AL 的内容

EXIT:

ENDM

调用宏指令的代码:

Data segment

Array db 'This is example!'

Data ends

Code segment

Assume cs:code,ds:data

Start:

Mov ax,data

Mov ds,ax

CLRB 16,array

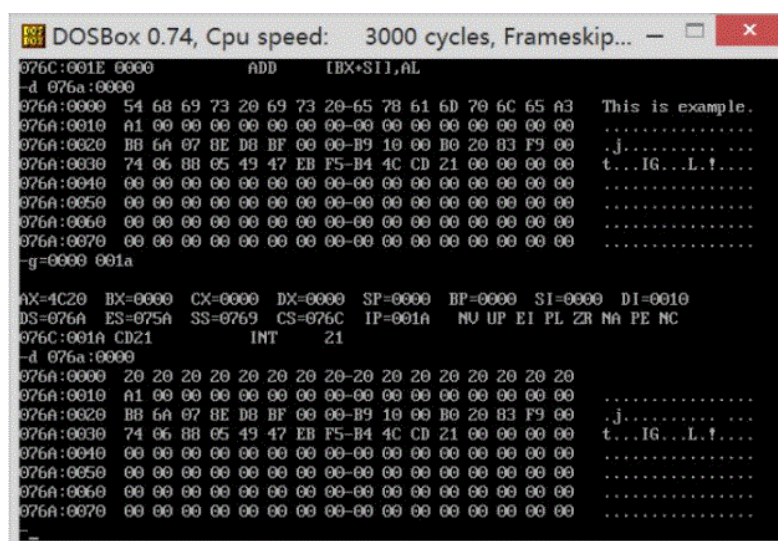
Mov ah,4ch

Int 21h

Code ends

End start

实验结果如图一所示:



图一 空格取代字符

2、BIN_SUB 完成多个字节数据连减的功能:

RESULT \leftarrow (A-B-C-D-...)

要相减的字节数据顺序存放在首地址为 OPERAND 的数据区中，减数的个数存放在 COUNT 单元中，最后结果存入 RESULT 单元。

代码如下：

宏指令：

```
BIN_SUB    MACRO    OPERAND, COUNT, RESULT

    PUSH    AX
    PUSH    BX
    PUSH    CX
    PUSH    DX
    PUSH    SI
    MOV     BX, OFFSET OPERAND
    MOV     CL, COUNT
    MOV     SI, OFFSET RESULT
    MOV     AL, [BX]
    INC     BX
    MOV     DL, [BX]
    SUB     AL, DL

COMP:
    DEC     CL
    CMP     CL, 1
    JE      EXIT
    INC     BX
    MOV     DL, [BX]
    SUB     AL, DL
    JMP     COMP

EXIT:
    MOV     RESULT, AL
```

```

POP    SI
POP    DX
POP    CX
POP    BX
POP    AX

ENDM

```

调用宏指令的代码：

Data segment

Array db 155,10,20,30

Count db 4

Result db ?

Data ends

Code segment

Assume cs:code,ds:data

Start:

Mov ax, data

Mov ds, ax

BIN_SUB array, count, result

Mov ah, 4ch

Int 21h

Code ends

End start

实验结果如图二所示：

图二 数组相减

The screenshot shows the DOSBox 0.74 interface with the following assembly code being executed:

```

076B:002F 5B      POP     BX
076B:0030 5B      POP     AX
076B:0031 B44C    MOV     AH,4C
076B:0033 CD21    INT     21
076B:0035 0000    ADD     [BX+SI],AL
076B:0037 0000    ADD     [BX+SI],AL
076B:0039 0000    ADD     [BX+SI],AL
076B:003B 0000    ADD     [BX+SI],AL
076B:003D 0000    ADD     [BX+SI],AL
076B:003F 0000    ADD     [BX+SI],AL
-g=0000 0033

```

Below the code, the register state is shown:

```

AX=4C6A BX=0000 CX=0000 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0033  NU UP EI PL ZR NA PE NC
076B:0033 CD21    INT     21
-d 076a:0000

```

The memory dump shows the contents of memory starting from 076A:0000:

```

076A:0000  9B 0A 14 1E 04 5F 00 00 00 00 00 00 00 00 00 00  .....
076A:0010  BB 6A 07 8E D0 50 53 51 52 56 BB 00 00 8A 0E 04  .j...PSQRU.....
076A:0020  00 BE 05 00 8A 07 43 8A 17 2A C2 EB F2 A2 05 00 5E 5A 59 5B  ....C...*.....
076A:0030  74 07 43 8A 17 2A C2 EB F2 A2 05 00 5E 5A 59 5B  t.C...*.....^ZYI
076A:0040  5B B4 4C CD 21 00 00 00 00 00 00 00 00 00 00 00  X.L.!.....
076A:0050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
076A:0060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
076A:0070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....

```