

廈門大學



《汇编语言》实验报告

(四)

姓 名 宋浩元

学 号 37220232203808

学 院 信息学院

专 业 软件工程

2024 年 11 月

1 实验目的

- (1) 基于已学习的第三章内容，学习如何运用标号、变量等伪指令完成程序设计。
- (2) 利用 DEBUG 调试程序，进一步理解各种伪指令的具体含义与作用。

2 实验环境

Windows11 环境下的 masm 与 DOSBOX;

3 实验内容

(1) 编程实现例 3.2 中的数据段定义，如图 1 所示，使用 debug 命令观察内存

状态，并回答下述问题：

a) 利用 9 号功能，将 msg 处的字符串输出，会出现什么情况？如果报错，应如何修改？

b) 若将数据段定义改成图 2 所示，请结合内存状态分析：

- dvar 开始存放的两个操作数有何不同，为什么？对于操作数

4294967295，数据定义为双字 dd 和 3 字 df，其在内存存放的状态有何不同？

- 若要满足 abc 开始定义的字符在内存中目前存放的状态，尝试将 dw 改为 db 伪指令，请给出修改后的具体伪指令代码

- bbuf 开始定义的字符串'day'，在内存中存放状态为何不同

- 若将 db 'xiamen university!' 改写成 dt 'xiamen', 'university'，结果如何？

- dbuf 开始的两条数据定义伪指令，实际运行时，内存分配有何不同？为什么？

图 1. 例 3.2 图 2

(2) 根据下述情况，分别编写程序，记录 BX 中 1 的个数（需要考虑 BX 中二进制串的特殊情况），要求如下：

- 循环次数已知
- 循环次数未知

(3) 按照下列要求，编写相应程序段。

- 1) 起始地址为 string 的主存单元中存放一个字符串（长度大于 6），把该字符串中的第 1 个和第 6 个字符（字节量）传送给 DX 寄存器；
 - 2) 从主存 buffer 开始的 4 字节中保存了 4 个非压缩 BCD 码，现按低（高）地址对低（高）位的原则，将他们合并到 DX 中。
 - 3) 假设从 B800H:0 开始存放有 100 个 16 位无符号数，编程求它们的和，并存在 DX.AX 中
 - 4) 一个 100 字节元素的数组首地址为 array，将每个元素减 1（不考虑溢出）。
- (4) 把内存中从 PACKED 开始的 10 个字节单元中的 20 位压缩 BCD 数转换成非压缩 BCD 数，并把结果存放在 UNPACKED 开始的 20 个字节单元中；将下列代码补充完整，并且自己定义 PACKED 中的数据，将 UNPACKED 中的结果展示出来。

```
MOV DX, _____  
MOV CL, _____  
MOV SI, 0  
MOV DI, ____  
CONVERT: MOV AL, [SI+PACKED]  
MOV AH, AL  
AND AL, 0FH  
_____  
MOV [DI+UNPACKED], ____  
ADD DI, _____
```

DEC DX

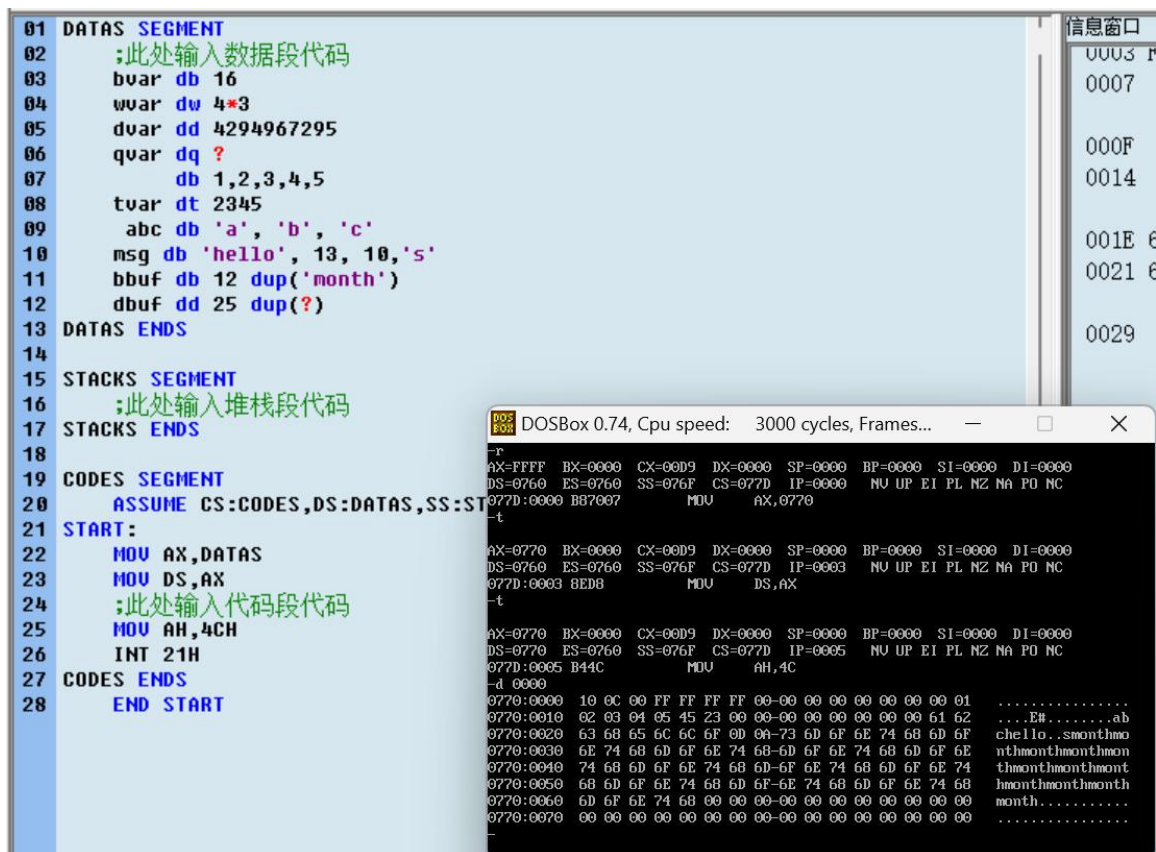
JNZ CONVERT

(5) 给定一个有序数组（均小于 FFH 例如 02H, 07H, 0BH, 0FH, 13H, 1CH, 24H, 39H, 40H, 57H, 68H）和一个目标值（例如 79H），请判断数组中是否含有两个数的和为目标值，请设计一个算法，将时间复杂度控制在 $O(n)$ ，编程实现并验证你的算法

4 实验具体实现

实验一：

(1) 观察内存状态：与定义一致。



The screenshot shows a DOSBox window with two panes. The left pane displays assembly code with line numbers 01 through 28. The right pane shows a memory dump with addresses and hex values, along with a text representation of the data.

```
01 DATAS SEGMENT
02 ;此处输入数据段代码
03 bvar db 16
04 wvar dw 4*3
05 dvar dd 4294967295
06 kvar dq ?
07 db 1,2,3,4,5
08 tvar dt 2345
09 abc db 'a', 'b', 'c'
10 msg db 'hello', 13, 10, 's'
11 bbuf db 12 dup('month')
12 dbuf dd 25 dup(?)
13 DATAS ENDS
14
15 STACKS SEGMENT
16 ;此处输入堆栈段代码
17 STACKS ENDS
18
19 CODES SEGMENT
20 ASSUME CS:CODES,DS:DATAS,SS:STACKS
21 START:
22 MOV AX,DATAS
23 MOV DS,AX
24 ;此处输入代码段代码
25 MOV AH,4CH
26 INT 21H
27 CODES ENDS
28 END START
```

Memory Dump (Address | Hex | ASCII):

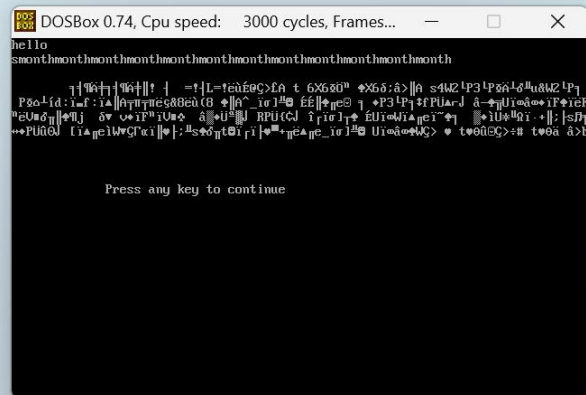
Address	Hex	ASCII
0003	FF	
0007	FF	
000F	FF	
0014	FF	
001E	FF	
0021	FF	
0029	FF	

(2) 如果直接调用会出现输出之后的乱码的情况

```

01 DATAS SEGMENT
02     ;此处输入数据段代码
03     buvar db 16
04     wvar dw 4*3
05     dvar dd 4294967295
06     quar dq ?
07     db 1,2,3,4,5
08     tvar dt 2345
09     abc db 'a', 'b', 'c'
10     msg db 'hello', 13, 10, 's'
11     bbuf db 12 dup('month')
12     dbuf dd 25 dup(?)
13 DATAS ENDS
14
15 STACKS SEGMENT
16     ;此处输入堆栈段代码
17 STACKS ENDS
18
19 CODES SEGMENT
20     ASSUME CS:CODES,DS:DATAS,SS:STACKS
21 START:
22     MOV AX,DATAS
23     MOV DS,AX
24     ;此处输入代码段代码
25     MOV AX, SEG msg
26     MOV DS, AX
27     MOV DX, OFFSET msg
28     MOV AH, 9
29     INT 21H
30     MOV AH,4CH
31     INT 21H
32 CODES ENDS
33     END START

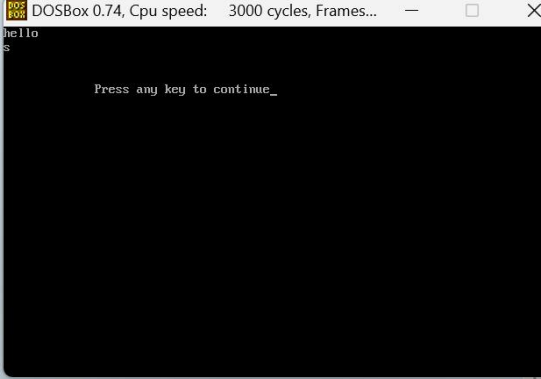
```



因为没有给 msg 添加字符串结束标志。

添加后如图显示：输出符合预期。

```
01 DATAS SEGMENT
02 ;此处输入数据段代码
03 bvar db 16
04 wvar dw 4*3
05 dvar dd 4294967295
06 qvar dq ?
07 db 1,2,3,4,5
08 tvar dt 2345
09 abc db 'a', 'b', 'c'
10 msg db 'hello', 13, 10, 's', '$'
11 bbuf db 12 dup('month')
12 dbuf dd 25 dup(?)
13 DATAS ENDS
14
15 STACKS SEGMENT
16 ;此处输入堆栈段代码
17 STACKS ENDS
18
19 CODES SEGMENT
20 ASSUME CS:CODES,DS:DATAS,SS:STACKS
21 START:
22 MOV AX,DATAS
23 MOV DS,AX
24 ;此处输入代码段代码
25 MOV AX, SEG msg
26 MOV DS, AX
27 MOV DX, OFFSET msg
28 MOV AH, 9
29 INT 21H
30 MOV AH,4CH
31 INT 21H
32 CODES ENDS
33 END START
```



(2)

(a) 用 `df` 存储结果为 `0000 ffff ffff` 和 `0000 0000 0001`。为三字存储。

用 `dd` 存储结果为 `ffff ffff` 和 `0000 0001`。为二字存储。

```

01 DATAS SEGMENT
02     bvar dw 16
03     wvar db 4*3
04     dvar df 4294967295
05     DF -4294967295
06     qvar dq ?
07     dw 1,2,3,4,5
08     tvar dt 2345
09     abc dw 'ab','c','b','cd'
10     msg db 'hello', 13, 10,'s'
11     bbuf df 'day'
12     db 'day'
13     db 'xiamen university!'
14     dbuf dq 10 dup('1234')
15     db 10 dup('1234')
16 DATAS ENDS
17
18 STACKS SEGMENT
19 ;此处输入堆栈段代码
20 STACKS ENDS
21
22 CODES SEGMENT
23     ASSUME CS:CODES,DS:DATAS,SS:STACKS
24 START:
25     MOV AX,DATAS
26     MOV DS,AX
27     MOV AH,4CH
28     INT 21H
29 CODES ENDS
30     END START
31

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frames...

IP	AX	CX	DX	SP	BP	SI	DI
0770:0000	B87607						
MOV AX,0770							
0770:0003	8ED8						
MOV DS,AX							
0770:0005	B44C						
MOV AH,4C							

Memory dump (hex):

```

0770:0000 10 00 0C FF FF FF FF 00-00 01 00 00 00 FF FF 00 .....
0770:0010 00 00 00 00 00 00 00 01-00 02 00 03 00 04 00 05 .....
0770:0020 00 45 23 00 00 00 00 00-00 00 00 62 61 63 00 62 .Ea.....bae.b
0770:0030 00 64 63 68 65 6C 6C 6F-60 00 73 79 61 64 00 00 .dhello..syad..
0770:0040 00 64 61 79 78 69 61 6D-65 6E 20 75 6E 69 76 65 .dayxiamen univ
0770:0050 72 73 69 74 79 21 34 33-32 31 00 00 00 00 34 33 rsity!4321....43
0770:0060 32 31 00 00 00 00 34 33-32 31 00 00 00 00 34 33 21....4321....43
0770:0070 32 31 00 00 00 00 34 33-32 31 00 00 00 00 34 33 21....4321....43

```

```

01 DATAS SEGMENT
02     bvar dw 16
03     wvar db 4*3
04     dvar dd 4294967295
05     dd -4294967295
06     quar dq ?
07     dw 1,2,3,4,5
08     tvar dt 2345
09     abc dw 'ab','c','b','cd'
10     msg db 'hello', 13, 10,'s'
11     bbuf df 'day'
12     db 'day'
13     db 'xiamen university!'
14     dbuf dq 10 dup('1234')
15     db 10 dup('1234')
16 DATAS ENDS
17
18 STACKS SEGMENT
19     ;此处输入堆栈段代码
20 STACKS ENDS
21
22 CODES SEGMENT
23     ASSUME CS:CODES,DS:DATAS,SS:STACKS
24 START:
25     MOV AX,DATAS
26     MOV DS,AX
27     MOV AH,4CH
28     INT 21H
29 CODES ENDS
30     END START
31

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frames...

```

AX=FFFF BX=0000 CX=0009 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0760 ES=0760 SS=076F CS=077D IP=0000 NU UP EI PL NZ Na PO NC
077D:0000 B87097 MOV AX,0770
-t
AX=0770 BX=0000 CX=0009 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0760 ES=0760 SS=076F CS=077D IP=0003 NU UP EI PL NZ Na PO NC
077D:0003 8ED8 MOV DS,AX
-t
AX=0770 BX=0000 CX=0009 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=077D IP=0005 NU UP EI PL NZ Na PO NC
077D:0005 B44C MOV AH,4C
-t
0770:0000 10 00 0C FF FF FF 01-00 00 00 00 00 00 00 00 .....
0770:0010 00 00 00 01 00 02 00 03-00 04 00 05 00 45 23 00 .....EM.
0770:0020 00 00 00 00 00 00 00 62-61 63 00 62 00 64 63 68 .....bac.b.dch
0770:0030 65 6C 6C 6F 00 0A 73 79-61 64 00 00 00 64 61 79 ello..syad...day
0770:0040 78 69 61 6D 65 6E 20 75-6E 69 76 65 72 73 69 74 xiamen universit
0770:0050 79 21 34 33 32 31 00 00-00 00 34 33 32 31 00 00 y4321....4321..
0770:0060 00 00 34 33 32 31 00 00-00 00 34 33 32 31 00 00 ..4321....4321..
0770:0070 00 00 34 33 32 31 00 00-00 00 34 33 32 31 00 00 ..4321....4321..

```

(b) 修改后的代码: `abc db 'a', 'b', 'c', 'b', 'c', 'd'`

(c) 可以看到 db 下的“abc”为 64 61 79.而 df 下的“abc”被当成一个 48 位的整数 00 00 00 79 61 64H


```

01 DATAS SEGMENT
02     bvar dw 16
03     wvar db 4*3
04     dvar dd 4294967295
05     dd -4294967295
06     qvar dq ?
07     dw 1,2,3,4,5
08     tvar dt 2345
09     abc dw 'ab','c','b','cd'
10     msg db 'hello', 13, 10, 's'
11     bbuf db 'day'
12     db 'day'
13     db 'xiamen university!'
14     dbuf dq 10 dup('1234')
15     db 10 dup('1234')
16 DATAS ENDS
17
18 STACKS SEGMENT
19     ;此处输入堆栈段代码
20 STACKS ENDS
21
22 CODES SEGMENT
23     ASSUME CS:CODES,DS:DATAS,SS:STACKS
24 START:
25     MOV AX,DATAS
26     MOV DS,AX
27     MOV AH,4CH
28     INT 21H
29 CODES ENDS
30     END START
31

```

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frames...
AX=FFFF BX=0000 CX=0000 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0760 ES=0760 SS=076F CS=077D IP=0000 NU UP EI PL NZ NA PO NC
077D:0000 B07007 MOV AX,0770
-t
AX=0770 BX=0000 CX=0000 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0760 ES=0760 SS=076F CS=077D IP=0003 NU UP EI PL NZ NA PO NC
077D:0003 8ED0 MOV DS,AX
-t
AX=0770 BX=0000 CX=0000 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=077D IP=0005 NU UP EI PL NZ NA PO NC
077D:0005 B44C MOV AH,4C
-t
-d 0000
0770:0000 10 00 0C FF FF FF FF 01-00 00 00 00 00 00 00 00 .....
0770:0010 00 00 00 01 00 02 00 03-00 04 00 05 00 45 23 00 .....EB.
0770:0020 00 00 00 00 00 00 00 62-61 63 00 62 00 64 63 68 .....bac.b.dch
0770:0030 65 6C 6C 6F 00 00 73 79-61 64 00 00 00 64 61 79 .....ello..syad...day
0770:0040 78 69 61 6D 65 6E 20 75-6E 69 76 65 72 73 69 74 .....xiamen universit
0770:0050 79 21 34 33 32 31 00 00-00 00 34 33 32 31 00 00 .....y14321....4321...
0770:0060 00 00 34 33 32 31 00 00-00 00 34 33 32 31 00 00 .....4321....4321...
0770:0070 00 00 34 33 32 31 00 00-00 00 34 33 32 31 00 00 .....4321....4321...
-t

```

(d) db 是定义字节数据的伪指令，dt 通常用于定义 10 字节的压缩 BCD 码。将 db 'xiamen university!' 改写成 dt 'xiamen', 'university' 是不合适的，因为 dt 不适合存储字符串。使用 dt 会导致数据存储和解释错误

我的程序
- 3.21.asm
- 3.24.asm
- 3.22.asm
- 3.15.asm
- 3.11.asm
- 3.3.asm
- 3.2.asm
- 4.1.asm
- 4.1(2).asm
软件应用问题解答
简单的入门程序实例
数据传送指令(免费)
算术运算指令
逻辑运算指令
串操作指令
程序转移指令
汇编伪指令
DOS汇编源程序
WINDOWS汇编实例源程序
汇编语言错误信息表
DOS 功能调用表(免费)
BIOS 功能调用表(免费)
DEBUG 命令(免费)
ASCII码字符表(免费)

```
01 DATAS SEGMENT
02     bvar dw 16
03     wvar db 4*3
04     dvar dd 4294967295
05     dd -4294967295
06     qvar dq ?
07     dw 1,2,3,4,5
08     twar dt 2345
09     abc dw 'ab','c','b','cd'
10     msg db 'hello', 13, 10,'s'
11     bbuf db 'day'
12     db 'day'
13     dt 'xiamen','university!'
14     dbuf dq 10 dup('1234')
15     db 10 dup('1234')
16 DATAS ENDS
17
18 STACKS SEGMENT
19     ;此处输入堆栈段代码
20 STACKS ENDS
21
22 CODES SEGMENT
23     ASSUME CS:CODES,DS:DATAS,SS:STACKS
24 START:
25     MOV AX,DATAS
26     MOV DS,AX
27     MOV AH,4CH
28     INT 21H
29 CODES ENDS
30     END START
31
```

【兼容WinXP模式】编译源程序 D:\asm_file\4.1(2).asm
D:\asm_file\4.1(2).asm(13): error A2008: syntax error : dq

【兼容WinXP模式】编译源程序 D:\asm_file\4.1(2).asm
D:\asm_file\4.1(2).asm(13): A2008 错误: dq: 语法错误 (免费使用)

(e) 在图 2 中，dbuf dq 10 dup('1234')和 db 10 dup('1234')定义了数据。dq 是定义四字（64 位）数据的伪指令，db 是定义字节数据的伪指令。使用 dq 10 dup('1234')时，每个'1234'会被存储为一个 64 位数据（8 字节），总共分配 80 字节（10 * 8）。使用 db 10 dup('1234')时，每个'1234'会被存储为 4 个字节（每个字符一个字节），总共分配 40 字节（10 * 4）。内存分配不同是因为 dq 和 db 定义的数据类型大小不同，dq 是 64 位（8 字节），db 是 8 位（1 字节）。

```

01 DATAS SEGMENT
02     bvar dw 16
03     wvar db 4*3
04     dvar dd 4294967295
05     DD -4294967295
06     quar dq ?
07     dw 1,2,3,4,5
08     tvar dt 2345
09     abc dw 'ab','c','b','cd'
10     msg db 'hello', 13, 10,'s'
11     bbuf df 'day'
12     db 'day'
13     db 'xiamen university!'
14     dbuf dq 10 dup('1234')
15     db 10 dup('1234')
16 DATAS ENDS
17
18 STACKS SEGMENT
19     ;此处输入堆栈段代码
20 STACKS ENDS
21
22 CODES SEGMENT
23     ASSUME CS:CODES,DS:DATAS,SS:STACKS
24 START:
25     MOV AX,DATAS
26     MOV DS,AX
27     MOV AH,4CH
28     INT 21H
29 CODES ENDS
30     END START
31

```

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frames...
-r
AX=FFFF BX=0000 CX=0009 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0760 ES=0760 SS=076F CS=077D IP=0000 NU UP EI PL NZ Na PO NC
077D:0000 B87007 MOV AX,0770
-t
AX=0770 BX=0000 CX=0009 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0760 ES=0760 SS=076F CS=077D IP=0003 NU UP EI PL NZ Na PO NC
077D:0003 8ED0 MOV DS,AX
-t
AX=0770 BX=0000 CX=0009 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=077D IP=0005 NU UP EI PL NZ Na PO NC
077D:0005 B44C MOV AH,4C
-t
-d 0000
0770:0000 10 00 0C FF FF FF 01-00 00 00 00 00 00 00 .....
0770:0010 00 00 00 01 00 02 00 03-00 04 00 05 00 45 23 00 .....EM.
0770:0020 00 00 00 00 00 00 00 62-61 63 00 62 00 64 63 68 .....bac.b.dch
0770:0030 65 6C 6C 6F 00 0A 73 79-61 64 00 00 00 64 61 79 ello..syad...day
0770:0040 78 69 61 6D 65 6E 20 75-6E 69 76 65 72 73 69 74 xiamen universit
0770:0050 79 21 34 33 32 31 00 00-00 00 34 33 32 31 00 00 y4321....4321..
0770:0060 00 00 34 33 32 31 00 00-00 00 34 33 32 31 00 00 ..4321....4321..
0770:0070 00 00 34 33 32 31 00 00-00 00 34 33 32 31 00 00 ..4321....4321..

```

实验二：

已知循环次数：

```

DATAS SEGMENT
    COUNT DW 0 ; 用于记录BX中1的个数
DATAS ENDS

STACKS SEGMENT
    DW 100 DUP(?) ; 定义堆栈段空间
STACKS ENDS

CODES SEGMENT
    ASSUME CS:CODES,DS:DATAS,SS:STACKS
START:
    MOV AX,DATAS
    MOV DS,AX

    MOV BX, 0FFFFH ; 这里可替换为任意16位二进制数
    MOV CX, 16 ; 因为BX是16位,所以循环次数为16

    MOV COUNT, 0 ; 初始化计数器为0

COUNT_LOOP:
    TEST BX, 0001H ; 测试最低位是否为1
    JZ ZERO ; 如果为0,跳转到ZERO
    INC COUNT ; 如果为1,计数器加1

ZERO:
    ROR BX, 1 ; 循环右移BX,准备测试下一位
    DEC CX
    JNZ COUNT_LOOP

    MOV AH, 4CH
    INT 21H

CODES ENDS
    END START

```

未知循环次数:

```

01 DATAS SEGMENT
02     COUNT DW 0 ; 用于记录BX中1的个数
03 DATAS ENDS
04
05 STACKS SEGMENT
06     DW 100 DUP(?) ; 定义堆栈段空间
07 STACKS ENDS
08
09 CODES SEGMENT
10     ASSUME CS:CODES,DS:DATAS,SS:STACKS
11 START:
12     MOV AX,DATAS
13     MOV DS,AX
14
15     MOV BX, 0FFFFH ; 这里可替换为任意16位二进制数
16
17     MOV COUNT, 0 ; 初始化计数器为0
18
19 COUNT_LOOP:
20     TEST BX, 0001H ; 测试最低位是否为1
21     JZ ZERO ; 如果为0, 跳转到ZERO
22     INC COUNT ; 如果为1, 计数器加1
23
24 ZERO:
25     ROR BX, 1 ; 循环右移BX, 准备测试下一位
26     CMP BX, 0 ; 检查BX是否为0
27     JNZ COUNT_LOOP
28
29     MOV AH, 4CH
30     INT 21H
31
32 CODES ENDS
33     END START

```

实验三：

```
DATAS SEGMENT
    string DB 'This is a sample string', 0 ; 示例字符串, 可替换为任意长度大于6的字符串
    len EQU $ - string ; 计算字符串长度
DATAS ENDS

STACKS SEGMENT
    DW 100 DUP(?)
STACKS ENDS

CODES SEGMENT
    ASSUME CS:CODES,DS:DATAS,SS:STACKS
START:
    MOV AX,DATAS
    MOV DS,AX

    MOV DL, string ; 传送第1个字符到DL
    MOV DH, string + 5 ; 传送第6个字符到DH

    MOV AH, 4CH
    INT 21H

CODES ENDS
    END START
```

```

01 DATAS SEGMENT
02     buffer DB 05H, 08H, 02H, 07H
03 DATAS ENDS
04
05 STACKS SEGMENT
06     DW 100 DUP(?)
07 STACKS ENDS
08
09 CODES SEGMENT
10     ASSUME CS:CODES,DS:DATAS,SS:STACKS
11 START:
12     MOV AX,DATAS
13     MOV DS,AX
14
15     MOV AL, buffer ; 取低地址的非压缩BCD码（低位）
16     MOV AH, buffer + 3 ; 取高地址的非压缩BCD码（高位）
17
18     MOV DX, 0
19     SHL AH, 4 ; 将高地址的BCD码移到高4位
20     OR DX, AX ; 合并到DX
21
22     MOV AH, 4CH
23     INT 21H
24
25 CODES ENDS
26     END START

```

```
01 DATAS SEGMENT
02
03 DATAS ENDS
04
05 STACKS SEGMENT
06     DW 100 DUP(?)
07 STACKS ENDS
08
09 CODES SEGMENT
10     ASSUME CS:CODES,DS:DATAS,SS:STACKS
11 START:
12     MOV AX, 0
13     MOV DX, 0
14     MOV AX, 0B800H
15     MOV ES, AX
16     MOV SI, 0
17     MOV CX, 100
18
19 SUM_LOOP:
20     MOV AX, ES:[B800H + SI] ; 取一个16位无符号数
21     ADD DX, AX ; 加到DX.AX中
22     ADC DX, 0 ; 处理进位
23     ADD SI, 2
24     DEC CX
25     JNZ SUM_LOOP
26
27     MOV AH, 4CH
28     INT 21H
29
30 CODES ENDS
31     END START
32
```



```
DATAS SEGMENT
    array DB 100 DUP(?)
DATAS ENDS

STACKS SEGMENT
    DW 100 DUP(?)
STACKS ENDS

CODES SEGMENT
    ASSUME CS:CODES,DS:DATAS,SS:STACKS
START:
    MOV AX,DATAS
    MOV DS,AX

    MOV CX, 100

ARRAY_LOOP:
    DEC BYTE PTR [array + CX - 1]
    DEC CX
    JNZ ARRAY_LOOP

    MOV AH, 4CH
    INT 21H

CODES ENDS
    END START
```

实验四：预期结果为 2501250125012501.一致。

```

DATAS SEGMENT
    PACKED DB 52h, 10h, 52h, 10h, 52h, 10h, 52h, 10h, 52h, 10h ; 示例的20位压缩BCD数, 可自行修改
    UNPACKED DB 20 DUP(?)
    MSG DB 'The unpacked BCD numbers are: $'
DATAS ENDS

STACKS SEGMENT
    DW 100 DUP(?)
STACKS ENDS

CODES SEGMENT
    ASSUME CS:CODES,DS:DATAS,SS:STACKS
START:
    MOV AX,DATAS
    MOV DS,AX

    MOV DX, 10 ; 循环次数, 因为要处理10个字节单元的压缩BCD数
    MOV CL, 4 ; 每次处理4位 (压缩BCD数特性)
    MOV SI, 0
    MOV DI, 0

CONVERT:
    MOV AL, [SI + PACKED]
    MOV AH, AL
    AND AL, 0FH ; 取低4位作为非压缩BCD数的低位
    MOV [DI + UNPACKED], AL
    ADD DI, 1 ; 结果存储单元偏移量增加1
    SHR AH, CL ; 将高4位移动到低4位
    MOV [DI + UNPACKED], AH ; 存储高4位作为非压缩BCD数的高位
    ADD DI, 1 ; 结果存储单元偏移量再增加1
    ADD SI, 1 ; 源数据单元偏移量增加1
    DEC DX
    JNZ CONVERT

    MOV AH, 9
    MOV DX, OFFSET MSG
    INT 21H

    MOV CX, 20
    MOV SI, 0

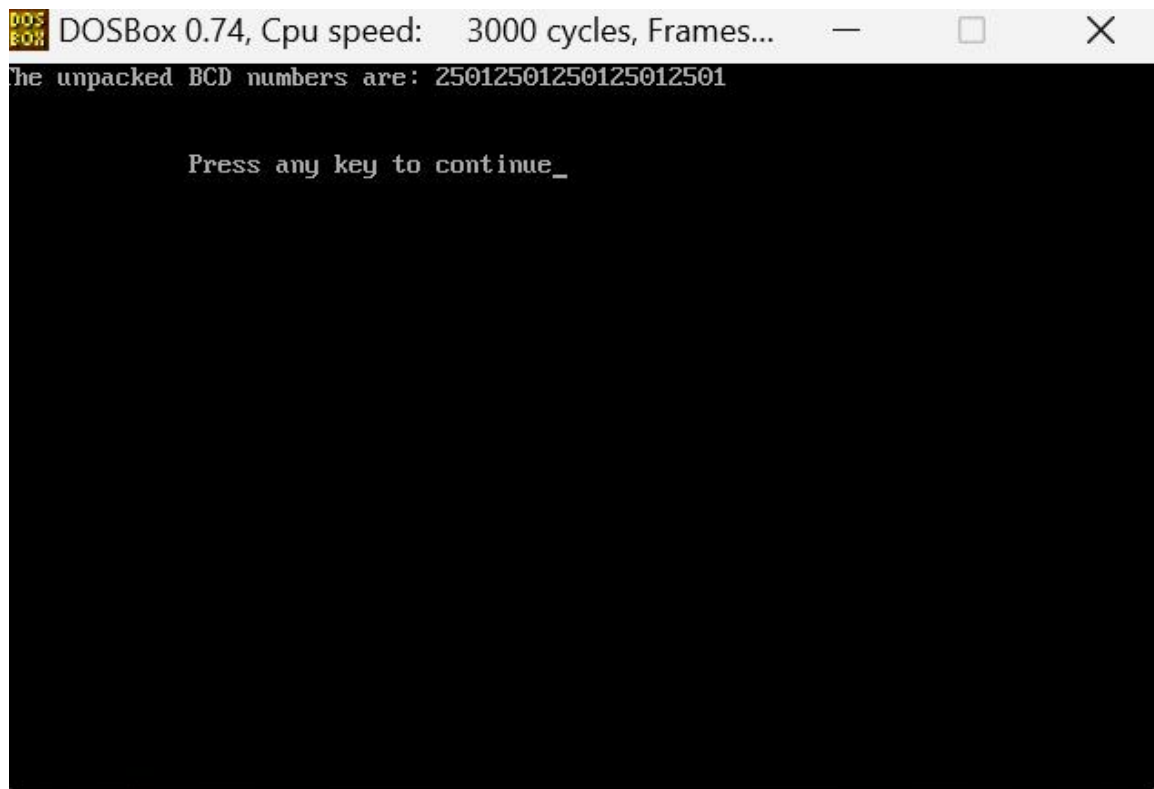
OUTPUT:
    MOV DL, [SI + UNPACKED]
    ADD DL, 20h

```

```

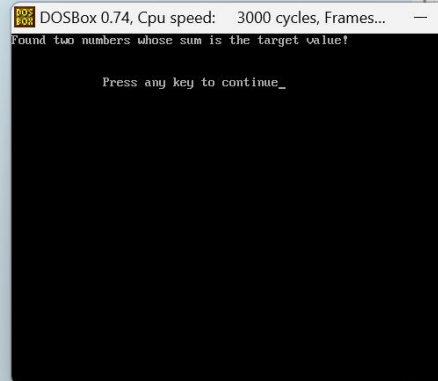
10
11 CODES SEGMENT
12     ASSUME CS:CODES,DS:DATAS,SS:STACKS
13 START:
14     MOV AX,DATAS
15     MOV DS,AX
16
17     MOV DX, 10 ; 循环次数, 因为要处理10个字节单元的压缩BCD数
18     MOV CL, 4 ; 每次处理4位 (压缩BCD数特性)
19     MOV SI, 0
20     MOV DI, 0
21
22 CONVERT:
23     MOV AL, [SI + PACKED]
24     MOV AH, AL
25     AND AL, 0FH ; 取低4位作为非压缩BCD数的低位
26     MOV [DI + UNPACKED], AL
27     ADD DI, 1 ; 结果存储单元偏移量增加1
28     SHR AH, CL ; 将高4位移动到低4位
29     MOV [DI + UNPACKED], AH ; 存储高4位作为非压缩BCD数的高位
30     ADD DI, 1 ; 结果存储单元偏移量再增加1
31     ADD SI, 1 ; 源数据单元偏移量增加1
32     DEC DX
33     JNZ CONVERT
34
35     MOV AH, 9
36     MOV DX, OFFSET MSG
37     INT 21H
38
39     MOV CX, 20
40     MOV SI, 0
41 OUTPUT:
42     MOV DL, [SI + UNPACKED]
43     ADD DL, 30H
44     MOV AH, 2
45     INT 21H
46     ADD SI, 1
47     DEC CX
48     JNZ OUTPUT
49
50     MOV AH, 4CH
51     INT 21H
52

```



实验五：用双指针来做这道两数之和是容易的。

```
01 DATAS SEGMENT
02     array DB 02H, 07H, 0BH, 0FH, 13H, 1CH, 24H, 39H, 40H, 57H, 68H
03     target DB 79H
04     len EQU $ - array
05     FOUND_MSG DB 'Found two numbers whose sum is the target value!$'
06     NOT_FOUND_MSG DB 'No two numbers in the array sum to the target value!$'
07 DATAS ENDS
08
09 STACKS SEGMENT
10     DW 100 DUP(?)
11 STACKS ENDS
12
13 CODES SEGMENT
14     ASSUME CS:CODES,DS:DATAS,SS:STACKS
15 START:
16     MOV AX,DATAS
17     MOV DS,AX
18
19     MOV SI, 0 ; 头指针, 指向数组起始位置
20     MOV DI, len - 1 ; 尾指针, 指向数组末尾位置
21
22 FIND_SUM:
23     MOV AL, array[SI]
24     MOV AH, array[DI]
25     ADD AL, AH ; 计算两数之和
26
27     CMP AL, target ; 比较两数之和与目标值
28     JE FOUND ; 如果相等, 说明找到了
29     JL INCREMENT_START ; 如果小于目标值, 头指针向后移动
30     JG DECREMENT_END
31
32 INCREMENT_START:
33     INC SI
34     JMP FIND_SUM
35
36 DECREMENT_END:
37     DEC DI
38     JMP FIND_SUM
39
40 FOUND:
41     MOV AH, 9
42     MOV DX, OFFSET FOUND_MSG
43     INT 21H
44     JMP EXIT
```



5 实验分析与总结

- (1) 学习了不同定义方式下存储方式和结果的不同，要适当的选择定义方式。
- (2) 巩固了循环，位移运算以及寻址方式。
- (3) 对汇编程序的整体结构有更深入的理解。