

实验三 简单编程练习

学号：61822313 姓名：钟锦程 实验日期：2024 年 4 月 15 日

1 实验任务和实验结果

1.1 实验任务 1

在一个数据块中找出最大数。假设数据块中的数据为 22、46、32、72、84、16、156，数据块的长度存放在 CX 寄存器中。

1. 数据块中的数据为无符号数，找出其中的最大数存放在以 MAXN1 为符号的单元中。
2. 数据块中的数据为有符号数，找出其中的最大数存放在以 MAXN2 为符号的单元中。

1.1.1 调试通过的源程序

```
1  DATA SEGMENT ;定义数据段
2  ARRAY DB 22,46,32,72,84,16,156 ;定义一串数据
3  MAXN1 DB 0 ;将存放最大值的变量初始化为0
4  DATA ENDS
5  CODE SEGMENT ;定义代码段
6      ASSUME DS:DATA,CS:CODE ;说明代码段、数据段
7  START:
8      MOV AX,DATA
9      MOV DS,AX ;给DS赋初值
10     MOV CX,6 ;置循环控制数
11     LEA DI,ARRAY ;将ARRAY表示的偏移地址送到DI
12     MOV DL,[DI] ;将第一个操作数送到寄存器中
13     JCXZ LAST
14 AGIN:
15     INC DI
16     CMP DL,[DI]
17     JAE NEXT ;使用无符号数比较指令JAE
18     MOV DL,[DI]
19 NEXT:
20     LOOP AGIN
```

```

21 LAST:
22     MOV MAXN1,DL
23     MOV AH,4CH
24     INT 21H
25 CODE ENDS
26     END START

```

1.1.2 实验结果

通过 Turbo 查看 MAXN1 相应的内存单元 (DS:0007)，发现其存储的内容是 9C，即无符号数 156，说明已经将最大数存到指定位置。

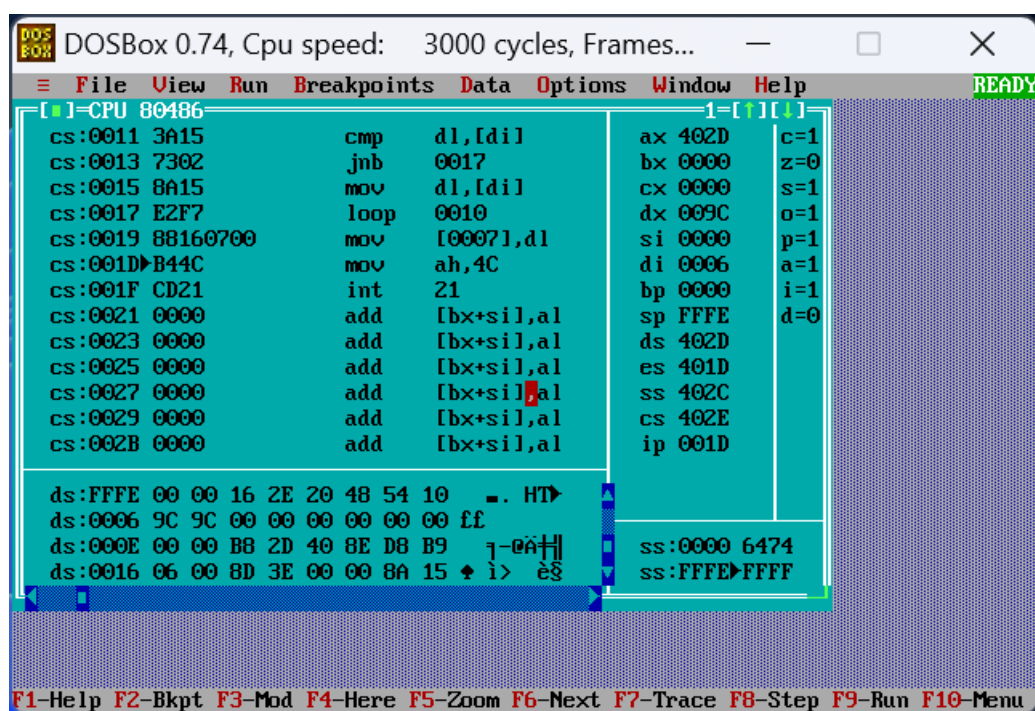


图 1: 实验 1（第一部分）结果截图

接下来是有符号数的比较

```

1 DATA SEGMENT ;定义数据段
2 ARRAY DB 22,46,32,72,84,16,156 ;定义一串数据
3 MAXN1 DB 0 ;将存放最大值的变量初始化为0
4 DATA ENDS
5 CODE SEGMENT ;定义代码段
6     ASSUME DS:DATA,CS:CODE ;说明代码段、数据段
7 START:
8     MOV AX,DATA
9     MOV DS,AX ;给DS赋初值
10    MOV CX,6 ;置循环控制数

```

```

11      LEA DI,ARRAY ;将ARRAY表示的偏移地址送到DI
12      MOV DL,[DI] ;将第一个操作数送到寄存器中
13      JCXZ LAST
14  AGIN:
15      INC DI
16      CMP DL,[DI]
17      JGE NEXT ;带符号数比较, DL>[DI]时跳转, 跳过更新环节
18      MOV DL,[DI]
19  NEXT:
20      LOOP AGIN
21  LAST:
22      MOV MAXN1,DL
23      MOV AH,4CH
24      INT 21H
25  CODE ENDS
26      END START

```

1.1.3 实验结果

通过 Turbo 查看 MAXN1 相应的内存单元 (DS:0007), 发现其存储的内容是 54, 即有符号数 84, 说明已经将最大数存到指定位置。

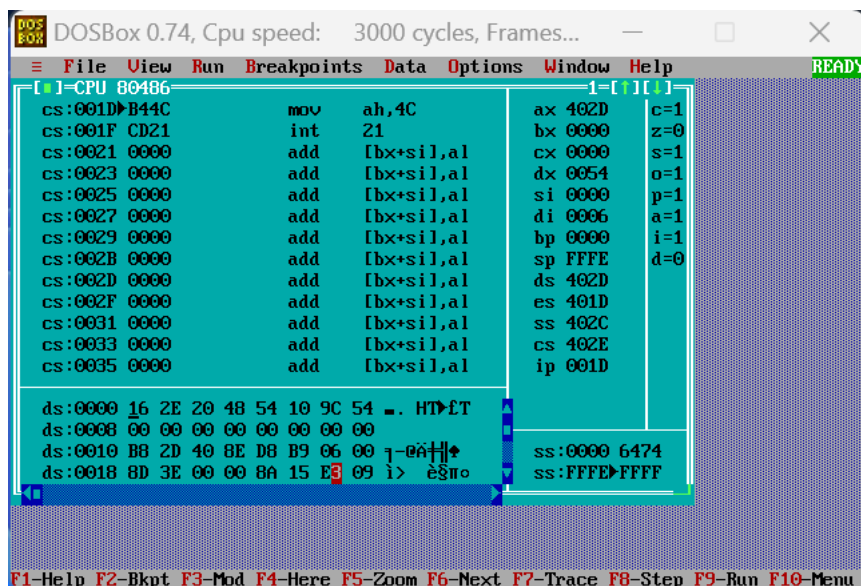


图 2: 实验 1 (第二部分) 结果截图

1.2 实验任务 3

1.2.1 实验任务的具体内容

求无符号字节数据之和，和数为 16 位二进制数。假设有数据 58、25，45，73、64，43，数据块的长度存放在 CX 寄存器中，和数存放在以 SUM 为符号的字单元中。

1.2.2 调试通过的源程序

```
1  DATA SEGMENT ;定义数据段
2  ARRAY DW 58,25,45,73,64,43 ;定义一串数据
3  SUM DW 0 ;将存放最大值的变量初始化为0
4  DATA ENDS
5  CODE SEGMENT
6  ASSUME DS:DATA,CS:CODE ;说明代码段、数据段
7  START:
8      MOV AX,DATA
9      MOV DS,AX ;给DS赋初值
10     MOV AX,0;
11     MOV CX,6
12     LEA DI,ARRAY ;将ARRAY表示的偏移地址送到DI
13     AGIN:
14         ADD AX,[DI];刚开始没加方框
15         INC DI
16         INC DI
17         LOOP AGIN ;该条指令执行后CX自动-1，手动写反而出错
18     LAST:
19         MOV SUM,AX
20         MOV AH,4CH
21         INT 21H
22     CODE ENDS
23         END START
24 ;当以CX=1状态来到LOOP时，LOOP不会继续循环，而是将CX-1然后跳出循环
```

1.2.3 实验结果

因为每个数为 16 位（2Byte），所以和数 SUM 的内存单元偏移地址为 000CH 和 000DH 两个字节。查看该内存单元内容，如图所示为 0134H，经检验，计算正确。

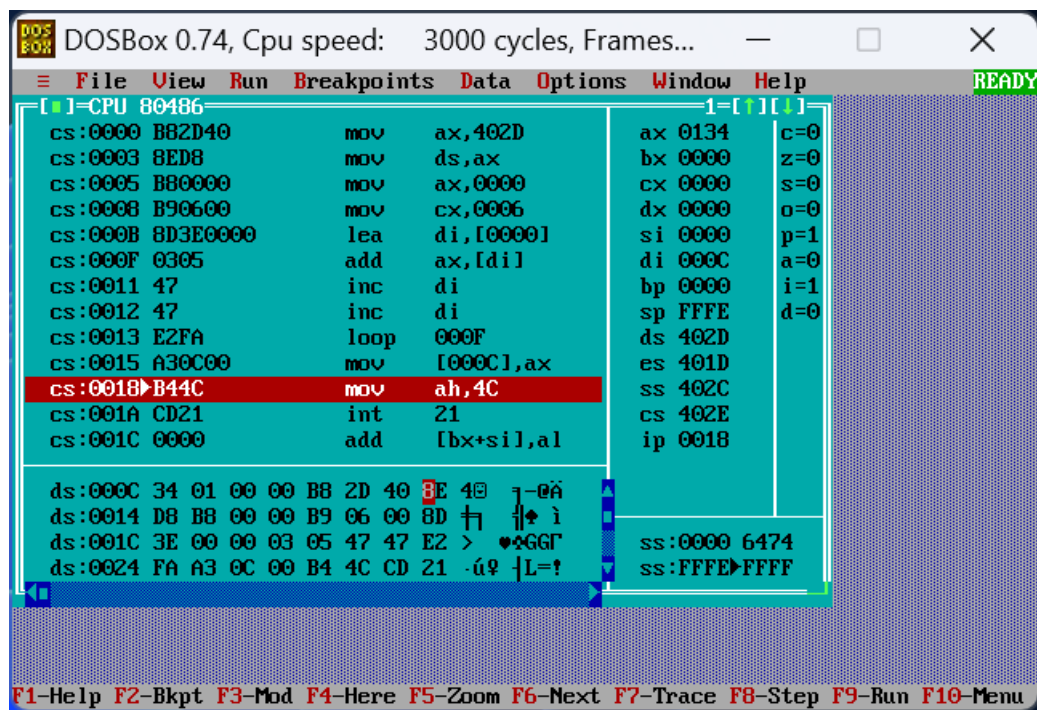


图 3: 实验 3 结果截图

1.3 实验任务 4

1.3.1 实验任务的具体内容

求两个十进制数相乘的积 ($56093 \times 5 = ?$) 改为 ($53348 \times 9 = ?$), 被乘数和乘数均以非压缩 BCD 码表示, 并存放在内存中, 乘积以非压缩 BCD 码的格式存放在以 SUM 为起始符号的单元中。

1.3.2 调试通过的源程序

```

1  DATA SEGMENT ;定义数据段
2  D1 DB 08,04,03,03,05
3  D2 DB 09 ;这两个十进制数相乘, 不超过6位
4  SUM DB 6 DUP(0) ;将存放最大值的变量初始化为0,预先不知道结果多长, 但是不会超
   过6位
5  DATA ENDS
6  CODE SEGMENT
7  ASSUME DS:DATA,CS:CODE ;说明代码段、数据段
8  GO:
9      MOV AX,DATA
10     MOV DS,AX ;给DS赋初值
11     MOV SI,OFFSET D1
12     MOV BL,[D2];;把乘数放入BL

```

```
13      MOV DI,OFFSET SUM ;目标存放地址放入DI
14      MOV DX,0;因后续进位存放在DL, 先清零
15      MOV CX,5 ;做乘法的次数,设置为SUM单元的长度
16      NEXT:
17      MOV AL,[SI] ;将被乘数从低位移入
18      INC SI
19      MUL BL
20      AAM;乘法的ASCII调整,AX中存放非压缩乘积,乘积的低位在AL,高位在AH
21      ADD AL,DL ;加上进位
22      AAA ;作加法的ASCII调整,把可能的进位加到AH上
23      MOV DL,AH;低位向高位的进位要留起来
24      MOV [DI],AL ;低位存入目的存储单元
25      INC DI
26      LOOP NEXT
27      MOV [DI],AH
28      MOV AH,4CH
29      INT 21H
30      CODE ENDS
31      END GO
```

1.3.3 实验结果

一个 5 位数乘以一个 1 位数,结果的长度不会超过 6 位,所以预先将答案对应的内存单元预置为 6Byte。存乘数时,低位排在前面,高位排在后面,从低位开始,每做一次乘法,都做一次 ASCII 调整,本位存入内存单元,进位暂时放在寄存器,作为下一次乘法的进位。最高位都做完乘法之后,也存入存储单元。结果如图

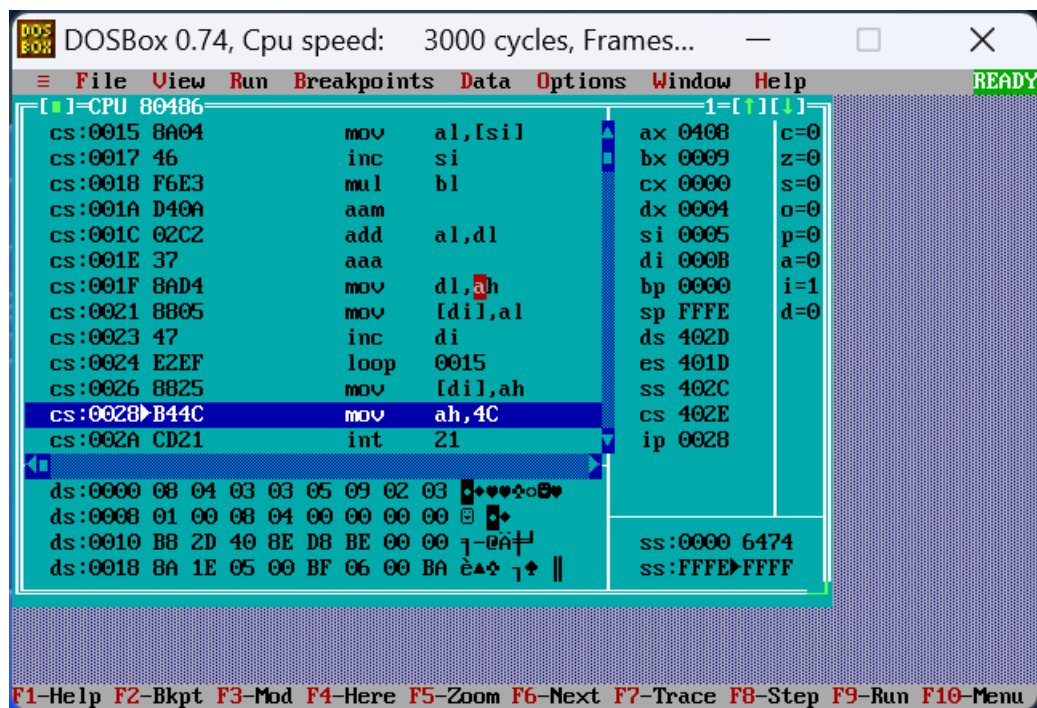


图 4: 实验 4 结果截图

结果的存储位置从 ds:0006H 开始，到 ds:000BH 结束。低位排在前面，每一位都是非压缩的 BCD 码。计算结果为 480132，经检验，计算正确。

1.4 实验任务 5

1.4.1 实验任务的具体内容

请用串传送指令编写程序，将以 STR1 为首地址的字节存储单元中的数据 30H、31H、32H、33H、34H、35H、36H、37H、38H、39H、40H、41H、42H、43H、44H、45H，传送到以 STR2 为首地址的字节存储单元中。

1.4.2 调试通过的源程序

```

1  DATA SEGMENT ;定义数据段
2  STR1 DB 30H,31H,32H,33H,34H,35H,36H,37H,38H,39H,40H,41H,42H,43H,44H,45H
3  STR2 DB 16 DUP(?)
4  DATA ENDS
5  CODE SEGMENT
6  ASSUME DS:DATA,CS:CODE ;说明代码段、数据段
7  GO:
8  MOV AX,DATA
9      MOV DS,AX ;给DS赋初值
10     MOV ES,AX ;给ES赋初值
11     MOV SI,OFFSET STR1

```



```
12      MOV DI,OFFSET STR2
13      MOV CX,16
14      CLD
15      REP MOVSB
16      MOV AH,4CH
17      INT 21H
18  CODE  ENDS
19      END GO
```

1.4.3 实验结果

数据串的每一个数据都是 Byte 为单位，所以用 MOVSB 指令。查看内存单元 ds:0010H 到 ds:0020H，可以看到数据串已经全部传送到指定位置

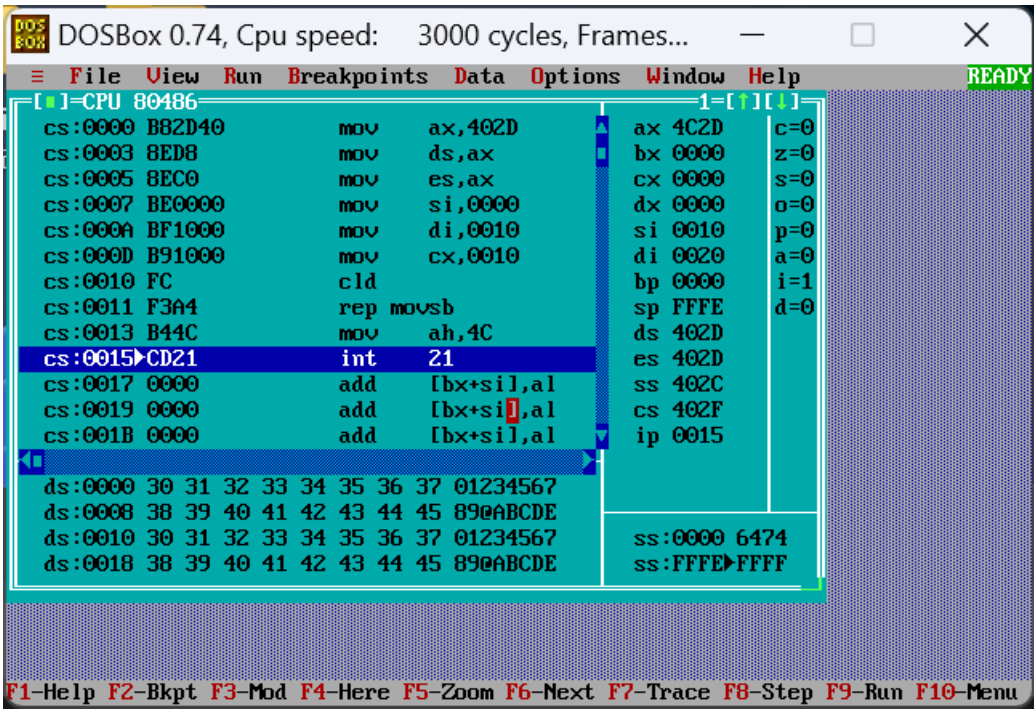


图 5: 实验 4 结果截图

1.5 实验任务 6

1.5.1 实验任务的具体内容

将任务 4 的乘积在屏幕上显示出来。

1.5.2 调试通过的源程序

```
1  DATA SEGMENT ;定义数据段
2  D1 DB 08,04,03,03,05
```



```
3      D2 DB 09 ;这两个十进制数相乘, 不超过6位
4      SUM DB 6 DUP(0) ;将存放最大值的变量初始化为0,预先不知道结果多长
5      OTHER DB 0DH,0AH,'$' ;为了使用字符串输出指令, 加上回车、换行和$
6      DATA ENDS
7      CODE SEGMENT
8      ASSUME DS:DATA,CS:CODE ;说明代码段、数据段
9      GO:
10         MOV AX,DATA
11         MOV DS,AX ;给DS赋初值
12         MOV SI,OFFSET D1
13         MOV BL,[D2];把乘数放入BL
14         MOV DI,OFFSET OTHER ;目标存放地址放入DI
15         DEC DI;指到SUM单元的最后一位, 低位放后面, 便于后续正序输出
16         MOV DX,0;因后续进位存放在DL, 先清零
17         MOV CX,5 ;做乘法的次数
18     NEXT:
19         MOV AL,[SI] ;将被乘数从低位移入
20         INC SI
21         MUL BL
22         AAM;乘法的ASCII调整,AX中存放非压缩乘积, 乘积的低位在AL, 高位在AH
23         ADD AL,DL ;加上进位
24         AAA ;作加法的ASCII调整, 把可能的进位加到AH上
25         MOV DL,AH;低位向高位的进位要留起来
26         ADD AL,30H;ASCII数字字符的范围是30H(0)~39H(9)
27         MOV [DI],AL ;低位存入目的存储单元
28         DEC DI
29         LOOP NEXT
30         ADD AH,30H;这时候已经是最后的高位了, 要加30H来从非压缩BCD码转换到
           ASCII码
31         MOV [DI],AH;此时DI已经指到最高位
32         CMP AH,30H
33         JNE LAST
34         INC DI ;若最高位为0.从次高位开始输出 (结果要么是5位要么是6位)
35     LAST:
36         MOV DX,DI; 字符串的首址放到DX, 准备输出
37         MOV AH,9 ;调用输出字符串的功能
38         INT 21H;输出字符串
39         MOV AH,4CH
40         INT 21H
```



```
10  CODE SEGMENT
11  ASSUME DS:DATA,CS:CODE,ES:EXTRA ;说明代码段、数据段
12  GO:
13      MOV CX,10
14      MOV AX,DATA
15      MOV DS,AX ;给DS赋初值
16      MOV AX,EXTRA
17      MOV ES,AX;给ES赋初值
18      MOV SI,OFFSET STR1;
19      MOV DI,OFFSET STR2
20      REPE CMPSB
21      JZ EQQ;若不完全一样，就不跳转
22      DEC SI;恢复到不一样的地方
23      MOV BX,SI;传送相异字节所在的地址
24      MOV [RESULT1],BL ;只有变量才可以用LOW指令
25      MOV AL,[SI]
26      MOV [RESULT2],AL
27      JMP LAST
28  EQQ:
29      MOV [RESULT1],0;若完全相同，RESULT1置0
30  LAST:
31      MOV AH,4CH
32      INT 21H
33  CODE ENDS
34      END GO
```

1.6.3 实验结果

第一次实验的结果如图所示，图8和图9展示的是 ES 段和 DS 段的数据，表明目的串与源串不同。图9显示了 DS 段的数据。从 ds:0000H 到 ds:0009H 存的是源串。ds:000AH 存的是 RESULT1，存放了第一个字符不一致的地址，即 06；ds:000BH 存的是 RESULT2，存放了源串中这个不一样的字符，即'2'。

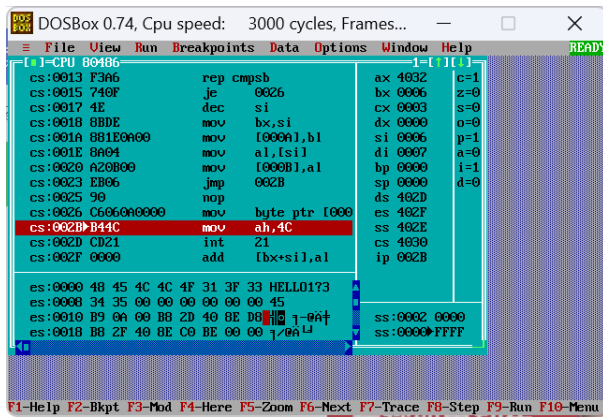


图 8: 实验 7 结果截图 1

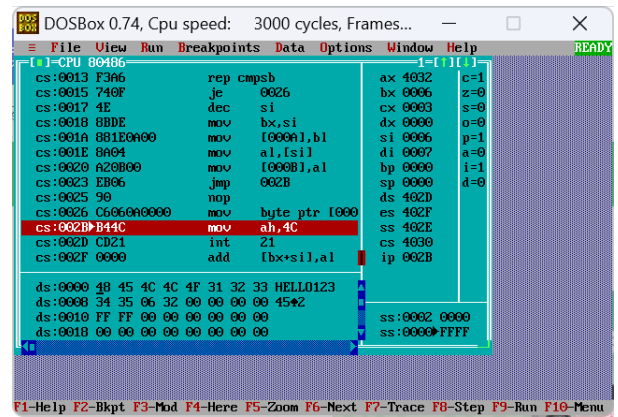


图 9: 实验 7 结果截图 2

第二次实验, 将两个字符串的内容设为一致, 都为'HELLO12345', 结果如图10、图11, 从后者可以看到, RESULT1 中存放的数据为 00H, 说明比较结果为两字符串相同。

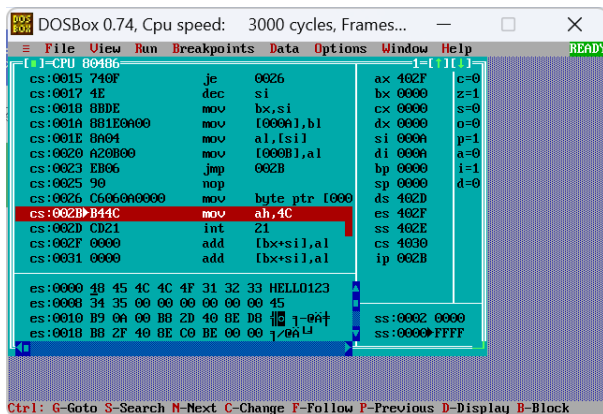


图 10: 实验 7 结果截图 3

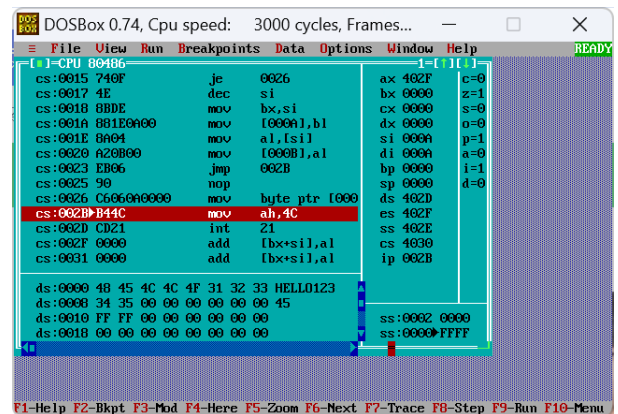


图 11: 实验 7 结果截图 4

2 实验总结

1. 实验 1 刚开始用字作为单位存储, 但每次 INC 只一次, 导致出错。后来发现可以用字节存储。
2. 将变量的地址装入寄存器时容易忘写 OFFSET
3. 实验 3 使用 MOV CX,(LENGTH DATA)-1; 置循环控制数。报错 must be associated with data。后来选择用立即数置数解决
4. 实验 3 用到 LOOP 指令, 该条指令执行后 CX 自动-1, 手动减少 CX 的值反而会出错
5. 实验 4: 预先不知道计算结果多长的时候可以先置足够大的空间备用。(在该环境下, 对没有用 DUP 定义的单元都认为 LENGTH=1?)

6. 定义字符串时注意全角/半角和中英文引号的区别
7. 实验 6 打印时刚开始总打印乱码。后来发现要将数字转化为 16 进制下的 ASCII 码才能正确打印。对非压缩 BCD 码，调整为 ASCII 只需要加 30H
8. 实验 7 最开始总是指不到不一样的那个字符。后来发现由于 CMPSB 指令使 SI 自增，因此找到相异处之后要将 SI-1 才指到不一样的那个字符

3 思考题

1. CMP BL,00 用于检查 BL 存的内容是否为 00H，可以用 SUB BL,00 代替。
2. 前者用 JA/JNBE 等，后者用 JGE/JNLE 等
3. 源操作数用的是间接寻址方式。可以用 MOV AX,NUM MOV SI,AX 两条指令代替
4. 用了 AAM 和 AAA 指令。调整方法：对于 AAA，若二进制数计算结果的低 4 位大于 9 或发生进位 (AF=1)，则要加上 0110 调整。若高四位大于 9 或发生进位 (CF=1)，也要加 0110 调整。对于 AAM，将 AL 的值除以 10H，结果放入 AH，AL 对 10H 取余，结果放在 AL。第一次执行 AAM 后 AX=0702H(十进制下 $8*9=72$)，第一次执行 AAA 后 AX=0403H(十进制下 $4*9+7=43$)
5. DF 用于控制串操作指令中，SI 和 DI 是自增还是自减。用 CLD 置 0，表示自增；用 STD 置 1，表示自减。