实验二　四则运算

实验报告

姓名：赵文亮

学号：2016011452

班级：自 64

日期：2018 年 4 月 9 日

目录

1 实验目的

1. 巩固 DEBUG 及宏汇编的使用。
2. 加深对运算指令的理解。
3. 注意标志寄存器的变化。

2 实验准备

1. **复习 DEBUG 命令及汇编程序上机操作**。
   - R：查看寄存器。标志位寄存器的缩写的含义如下：
     - OV = OVerflow, NV = No oVerflow. DN = DowN, UP (up).
     - EI = Enable Interupt, DI = Disable Interupt.
     - NG = NeGative, PL = PLus; a strange mixing of terms due to the fact that 'Odd Parity' is represented by PO (rather than POsitive), but they still could have used 'MI' for MInus.
     - ZR = ZeRo, NZ = Not Zero.
     - AC = Auxiliary Carry, NA = Not Auxiliary carry.
     - PE = Parity Even, PO = Parity Odd. CY = CarrY, NC = No Carry.
   - D：查看内存。
   - N：指定程序名称。
   - L：装载程序。
   - E：修改数据区。
   - G：执行程序。
   - P：单步执行。
   - T：执行并跟踪。
2. **按实验内容要求编写完整的源程序清单**。

   源程序和详细的分析见第 2 页第 4 节。

3 实验内容

本次实验的各个任务的实验步骤如下：

1. 编写汇编源程序（详见第 4 节）。
2. 编译、链接程序，生成可执行文件。
3. 进入 DEBUG 装载。

4. 查看反汇编结果。

5. 使用 D 命令查看数据区内存。

6. 使用 E 命令修改运算的原始数据。命令格式为：E[地址]。即可修改对应地址的数据，修改完一个数据后按下空格可以修改下一个字节中的数据，按下回车结束修改。

7. 运行程序。

8. 使用 D 命令查看数据区内存，验证运算结果。

9. 也可以使用单步执行命令逐行语句执行，以便观察寄存器的变化。

4 实验程序及说明

4.1 加减运算

4.1.1 十六进制

```
1  NAME MY_PROGRAM        ;程序模块名
2  DATA SEGMENT           ;数据段开始
3  NUM1     DB 0, 1
4  NUM2     DB 1, 1
5  BLK1     DB 12 DUP(0)
6  SUM DB 2 DUP(0)
7  DIF DB 2 DUP(0)
8  BLK2     DB 12 DUP(0)
9  MESS     DB 'HAVE DONE', 13, 10, '$'
10 DATA     ENDS           ;数据段结束
11 STACK    SEGMENT PARA STACK
12          DB 100 DUP(?)
13 STACK    ENDS           ;堆栈段结束
14 CODE     SEGMENT        ;代码段开始
15          ASSUME CS: CODE, DS: DATA, ES:DATA, SS:STACK
16
17 START:   MOV AX, DATA; init data
18          MOV DS, AX
19          MOV ES, AX
20
21          LEA SI, NUM1
22          MOV AX, [SI]
23          LEA SI, NUM2
24          MOV BX, [SI]
25
26          PUSH AX
27          ADD AL, BL
28          ADC AH, BH
```

```
29          LEA DI, SUM
30          MOV [DI], AX
31
32          POP AX
33          SUB AL, BL
34          SBB AH, BH
35          LEA DI, DIF
36          MOV [DI], AX
37
38          LEA DX, MESS      ;指向提示字符串
39          MOV AH, 9     ;显示字符串的功能号
40          INT 21H       ;DOS功能调用
41          MOV AH, 4CH ;退出用户程序的功能号
42          INT 21H       ;DOS功能调用
43  CODE    ENDS          ;代码段结束
44  END S   TART          ;整个源程序结束，并指明第一条执行语句
```

算法逻辑的流程图如图 1 所示。下面将代码主要部分进行分析。

- 2-10：数据段初始化。为两个操作数分配了内存，并给和、差分配内存。
- 17-19：初始化段基址。
- 21-24：将两个操作数分别存在 AX 和 BX 中。
- 26-30：计算两个操作数的和，并送存。这里使用了先计算低 8 位，再计算高 8 位（带进位）的步骤。其中 26 行中 PUSH AX 是因为在加法过程中 AX 的值会被破坏，所以需要提前入栈，以便在减法中恢复 AX 的值。
- 32-36：恢复 AX 的值，计算两个操作数的差并送存。类似与加法的步骤，按照从低 8 位到高 8 位的顺序。
- 38-42：显示提示信息、退出用户程序等操作。

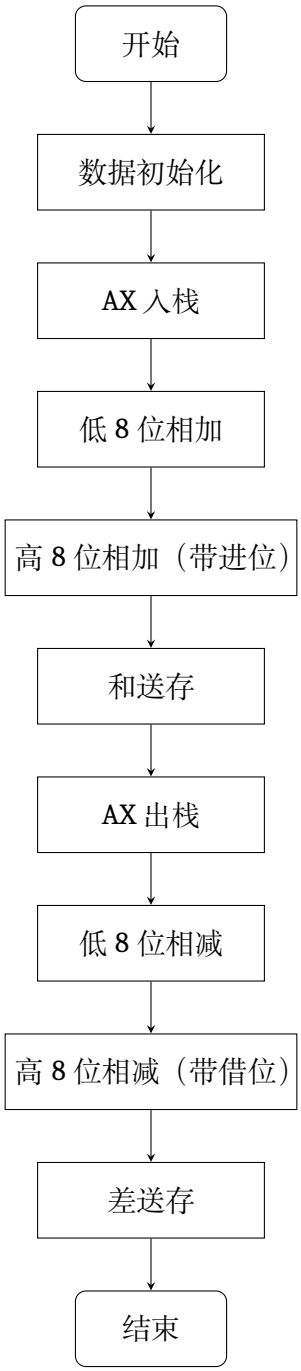

图 1: 十六进制加法流程图

4.1.2 十进制

```
NAME ADD_DEC;程序模块名
DATA SEGMENT          ;数据段开始
NUM1     DB 0, 1
NUM2     DB 1, 1
BLK1     DB 12 DUP(0)
SUM DB 2 DUP(0)
DIF DB 2 DUP(0)
BLK2     DB 12 DUP(0)
MESS     DB 'HAVE DONE', 13, 10, '$'
DATA     ENDS          ;数据段结束
STACK    SEGMENT PARA STACK
         DB 100 DUP(?)
STACK    ENDS          ;堆栈段结束
CODE     SEGMENT       ;代码段开始
         ASSUME CS: CODE, DS: DATA, ES:DATA, SS:STACK

START:   MOV AX, DATA; init data
         MOV DS, AX
         MOV ES, AX

         LEA SI, NUM1
         MOV AX, [SI]
         LEA SI, NUM2
         MOV BX, [SI]

         PUSH AX
         ADD AL, BL
         DAA
         ADC AH, BH
         PUSH AX
         MOV AL, AH
         DAA
         MOV AH, AL
         POP DX
         MOV AL, DL
         LEA DI, SUM
         MOV [DI], AX

         POP AX
         SUB AL, BL
         DAS
         SBB AH, BH
         PUSH AX
         MOV AL, AH
         DAS
         MOV AH, AL
```

```
47              POP DX
48              MOV AL, DL
49              LEA DI, DIF
50              MOV [DI], AX
51
52              LEA DX, MESS      ;指向提示字符串
53              MOV AH, 9     ;显示字符串的功能号
54              INT 21H       ;DOS功能调用
55              MOV AH, 4CH ;退出用户程序的功能号
56              INT 21H       ;DOS功能调用
57  CODE        ENDS          ;代码段结束
58  END         START         ;整个源程序结束，并指明第一条执行语句
```

十进制的 BCD 运算本质上和十六进制的差别不大，所不同的是在进行加减法之后需要分别使用 DAA 和 DAS 进行十进制调整。算法的流程图如图 2 。代码的主要分析如下：

- 2-10：初始化数据段。
- 17-19：初始化段基址。
- 21-24：保存操作数到 AX 和 BX 中。
- 26-37：计算 BCD 码的加法，并将结果送存。仍然分低 8 位和高 8 位进行计算。计算后，需要通过 DAA 进行十进制调整。需要指出的是，由于 DAA 命令是针对 AL 中的数据的，为了调整 AH 中的数据，我先将 AX 入栈，再将 AH 复制到 AL，调整结束后在送还。

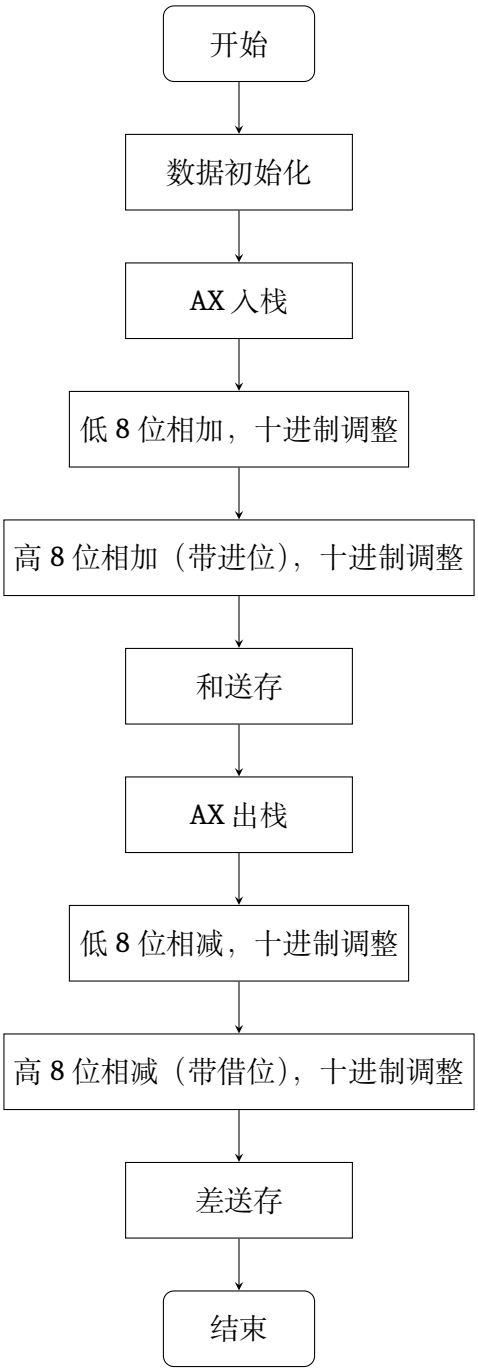
开始
数据初始化
AX 入栈
低 8 位相加，十进制调整
高 8 位相加（带进位），十进制调整
和送存
AX 出栈
低 8 位相减，十进制调整
高 8 位相减（带借位），十进制调整
差送存
结束

图 2: 十进制加法流程图

4.2 乘法运算

乘法运算的算法流程图取自实验指导书，BCD 码乘法和 ASCII 码乘法分别如图 3 和图 4 所示。

4.2.1 BCD 码

```
1  NAME MY_PROGRAM        ;程序模块名
2  DATA SEGMENT           ;数据段开始
3  NUM1     DB 23
4  NUM2     DB 32
5  BLK1     DB 14 DUP(0)
6  PRO      DB 2 DUP(0)
7  BLK2     DB 14 DUP(0)
8  MESS     DB 'HAVE DONE', 13, 10, '$'
9  DATA     ENDS           ;数据段结束
10 STACK    SEGMENT PARA STACK
11          DB 100 DUP(?)
12 STACK    ENDS           ;堆栈段结束
13 CODE     SEGMENT        ;代码段开始
14          ASSUME CS: CODE, DS: DATA, ES:DATA, SS:STACK
15
16 START:   MOV AX, DATA; init data
17          MOV DS, AX
18          MOV ES, AX
19
20          LEA SI, NUM1
21          MOV BL, [SI]
22          MOV CL, [SI+1]
23          MOV DX, 0H
24
25          CMP BL, 0
26          JE SAVE
27 LOOP:    CMP CL, 0
28          JE SAVE
29
30          MOV AL, DL; 低字节
31          ADD AL, BL
32          DAA
33          MOV DL, AL
34
35          MOV AL, DH; 高字节
36          ADC AL, 0
37          DAA
38          MOV DH, AL
39
40          MOV AL, CL
41          DEC AL
```

```
42              DAS
43              MOV CL, AL
44
45              JMP LOOP
46
47  SAVE:       LEA DI, PRO
48              MOV [DI], DX
49
50              LEA DX, MESS        ;指向提示字符串
51              MOV AH, 9       ;显示字符串的功能号
52              INT 21H         ;DOS功能调用
53              MOV AH, 4CH ;退出用户程序的功能号
54              INT 21H         ;DOS功能调用
55  CODE        ENDS            ;代码段结束
56  END         START           ;整个源程序结束，并指明第一条执行语句
```

- 2-10：数据段初始化。
- 16-18：段基址初始化。
- 20-23：将被乘数、乘数分别提取到 BL、CL 中，乘积 DX 清零。
- 25-26：如果 BL 为 0，则直接跳到积送存步骤。
- 27-45：如果 CL 不为 0，则不断循环：将 BL 中的数据和 DL 中的数据相加并做十进制调整，保存在 DL 中；CL 递减并做十进制调整。如果 CL 为 0，直接退出循环跳到送存步骤。这里所有的十进制调整都需要通过 AL 辅助完成。
- 47-48：积送存。
- 50-54：显示提示字符串，退出程序。

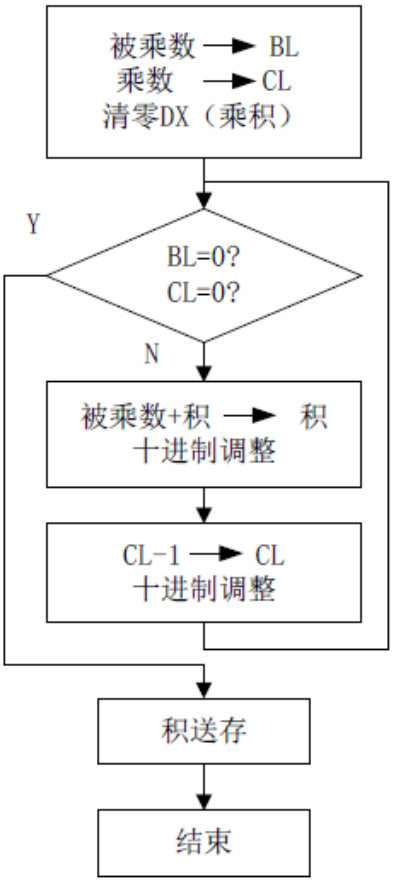

图 3: BCD 码乘法运算流程图

4.2.2 ASCII 码

ASCII 码的乘积多了 ASCII 和 BCD 码之间的相互转换的步骤。我通过将寄存器与'0' 相加或相减实现了从 BCD 到 ASCII 或从 ASCII 到 BCD 的转换。

```
1  NAME MY_PROGRAM       ;程序模块名
2  DATA SEGMENT          ;数据段开始
3  NUM1     DB '4', '5', '7', '2'
4  NUM2     DB '3'
5  BLK1     DB 11 DUP(0)
6  PRO      DB 4 DUP('0')
7  BLK2     DB 12 DUP(0)
8  MESS     DB 'HAVE DONE', 13, 10, '$'
9  DATA     ENDS          ;数据段结束
10 STACK    SEGMENT PARA STACK
11          DB 100 DUP(?)
12 STACK    ENDS          ;堆栈段结束
13 CODE     SEGMENT       ;代码段开始
14          ASSUME CS: CODE, DS: DATA, ES:DATA, SS:STACK
15
16 START:   MOV AX, DATA; init data
17          MOV DS, AX
18          MOV ES, AX
19
20          LEA SI, NUM2
21          MOV BL, [SI]
22          SUB BL, '0'
23          LEA SI, NUM1
24          LEA DI, PRO
25          MOV CX, 4
26
27 LOOP:    MOV AL, [SI]
28          SUB AL, '0'
29          MUL BL
30          AAM
31          INC SI
32          MOV DL, [DI]
33          SUB DL, '0'
34          ADD AL, DL
35          DAA
36          ADD AL, '0'
37          MOV [DI], AL
38          INC DI
39          ADD AH, '0'
40          MOV [DI], AH
41          DEC CX
42          JNZ LOOP
43
```

```
44              LEA DX, MESS      ;指向提示字符串
45              MOV AH, 9         ;显示字符串的功能号
46              INT 21H           ;DOS功能调用
47              MOV AH, 4CH       ;退出用户程序的功能号
48              INT 21H           ;DOS功能调用
49  CODE        ENDS              ;代码段结束
50  END         START             ;整个源程序结束，并指明第一条执行语句
```

- 2-9：数据段初始化。
- 16-28：段基址初始化。
- 20-25：数据初始化。
- 27-42：做四次循环：从 SI 取一位数字，通过减‘0’的方法得到对应的 BCD 码，并将其与 BL 中的数相乘，做十进制调整。将本位积与前一次积的进位相加，做十进制调整。本位积送存到 [DI]，本位积进位送存到 [DI+1]（送存前均转为 ASCII 码），递增 DI。
- 44-48：显示提示字符串，退出程序。

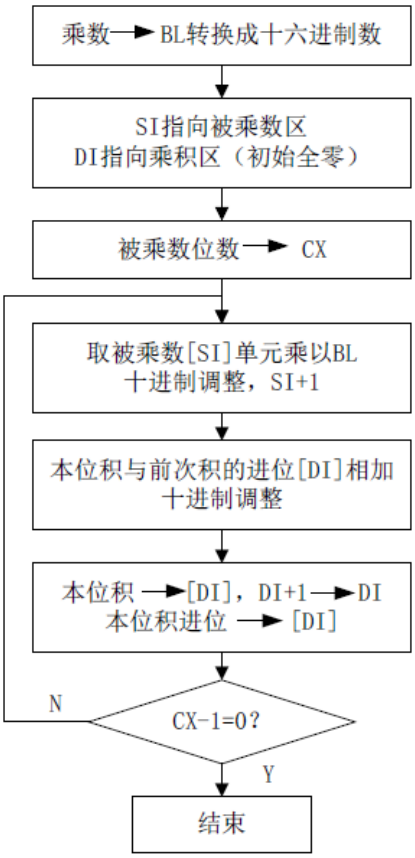

图 4: ASCII 码乘法运算流程图

5 完成情况和心得体会

在实验的预习中，我已经完成了所有程序并已经在自己的电脑上测试成功，所以我在机房的实验比较顺利，在很短的时间内就完成了所有内容。

本次实验中需要使用 E 命令修改内存，这是我之前没有尝试过的。不过通过与 D 命令类比，我很快就知道了 E 命令的用法，也尝试出了连续改动内存数据的方法，即改动完一个字节后按下空格。

本次实验的代码部分相比上次难度有所提升，涉及到的知识点也很多。除了一些基本的条件转移外，还需要各种运算以及十进制调整。一开始我对十进制调整命令不是很熟悉，不知道它的作用对象是什么。后来我认真翻阅了教材，发现十进制调整只可以对 AX 寄存器中的内容进行调整。在本实验中，由于使用压缩的 BCD 码，为了实现十进制调整，需要借助 AL 中转，调整结束后再送回原来的寄存器。掌握了这一个要点后，关于这方面的编程便显得不那么复杂了。

值得一提的是，我在中主上机的过程中，提前进行了一些测试，可是试了几次之后发现 DEBUG 程序开始出 bug。无奈只好重启电脑。后来课上老师说，在代码里最好不要写中文。我的代码里有一些中文注释，不知道是不是有所影响。下次实验我要更加注意这一点。

本次实验锻炼了我汇编编程的能力，也让我对之前的知识有了更加深刻的理解。感谢老师和助教的指导和付出！

参考文献

[1] 汇编中一些标志位的含义.https://blog.csdn.net/caoyuanll/article/details/50571498