厦門大學

《汇编语言》实验报告

（三）

姓　　名　宋浩元

学　　号　37220232203808

学　院　信息学院

专　业　软件工程

2024 年　10 月

1 实验目的

（1）利用 DEBUG 调试程序，理解和掌握第二章算术运算和逻辑运算类指令的工作原理以及对于标志位的影响；

（2）学会利用算数类指令完成简单的运算，掌握汇编语言的源程序结构；

（3）利用已学习的指令，完成简单的程序设计。

2 实验环境

Masm for Windows 继承环境

DOSBOX 0.74-3

3 实验内容

(1) 针对教材第二章中 2.4.2 条件转移指令中的例题 2.38,2.39, 2.43,2.44，若将其条件分支指令条件取反，请修改相应代码，完成要求的功能，并分析与原代码有何不同？（此部分不必提交）

(2) 用 DEBUG 单步调试下述指令，观察每条指令的运行结果以及标志寄存器 FLAG 的置位情况。

1) MOV AX,1470H
2) AND AX, AX
3) OR AX, AX
4) NOT AX
5) TEST AX,0F0F0H
6) MOV CL,4
7) MOV AL, 0F0H
8) SAR AL,1
9) SAR AL, CL
10) MOV CL,2
11) ROR AL, CL

12) MOV CL,4

13) RCL AL, 4

(3) 设 X,Y,Z,V 均为 16 位带符号数，分别存放在 X,Y,Z,V 存储单元中，部分代码如图 1 所示，请完成下述任务：

1) 自行在程序的数据定义部分对 X,Y,Z,V 进行赋值（要求：大于 100H 且 X,Y,Z,V 不相等）；

2) 调试程序，描述该程序实现什么功能？

3) 代码部分中，两次出现 CWD 指令，其作用各是什么？4) 最后的结果存放在哪里，结果为多少，为什么？

5) 给出第 5/13/19 行后的标志寄存器 FLAG 的状态，试说明原因

A.asm

```
1     MOV AX,DATAS
2     MOV DS,AX
3     MOV AX,WORD PTR X
4     MOV BX,WORD PTR Y
5     IMUL BX
6     MOV CX,AX
7     MOV BX,DX
8     MOV AX,WORD PTR Z
9     CWD
10    ADD CX,AX
11    ADC BX,DX
12    SUB CX,0540H
13    SBB BX,0
14    MOV AX,WORD PTR V
15    CWD
16    SUB AX,CX
17    SBB DX,BX
18    MOV BX,WORD PTR X
19    IDIV BX
20    MOV AH,4CH
21    INT 21H
```

(4) 根据图 2 所示程序段，回答下述问题：

1) 将 mov dx,0 语句替换成其他语句；

2) 改变被除数 NUMW 和除数（bx）的初始值，利用 debug 命令查看内存中存放结果与预期是否一致。3) 若操作数为有符号数，请问该段代码将如何修改？

4) 针对问题 3)，改变 NUMW 和 bx 的取值（正数，负数各取一组），根据运行结果，你会得出什么结论？

5）尝试将 bx 中的除数改为 0，观察编译和运行结果。

```
DATAS SEGMENT
   NUMW  dw  1000H ;存放16位无符号数，范围0~65535
   SHANG dw 0
   YUSHU dw 0      ;此处定义三个内存变量，NUMW被除数，SHANG，结果商，YUSHU，结果余数
DATAS ENDS

STACKS SEGMENT
   db 100 dup(?); 堆栈段
STACKS ENDS

CODES SEGMENT
    ASSUME CS:CODES,DS:DATAS,SS:STACKS
   ;代码段实现16位无符号数除法运算
START:
    MOV AX,DATAS
    MOV DS,AX
    mov ax, NUMW
    mov dx,0
    mov bx,10H    ; bx中存放除数
    div bx
    mov SHANG,ax
    mov YUSHU,dx
    MOV AH,4CH
    INT 21H
CODES ENDS
    END START
```

(5) 阅读图 3 中的程序段完成下述问题：

1) 该代码段实现什么功能（结合代码，给出具体表达式）？

2) 每次循环，寄存器 AX 和 CX 中存放的数值代表什么含义？

3) 循环体（again 为入口）执行了几次？若要改变循环次数，应修改哪条或者哪些指令？

```
START:
    MOV AX,DATAS
    MOV DS,AX
    mov cx,1
    mov bl,2
again:mov al,bl
     inc bl
     mul bl
     add cx,ax
     cmp ax,002AH
     jb again    ;无符号数比较指令，前者小于后者则转到again标号处执行
     MOV AH,4CH
     INT 21H
CODES ENDS
    END START
```

4 实验具体实现

实验（二）：

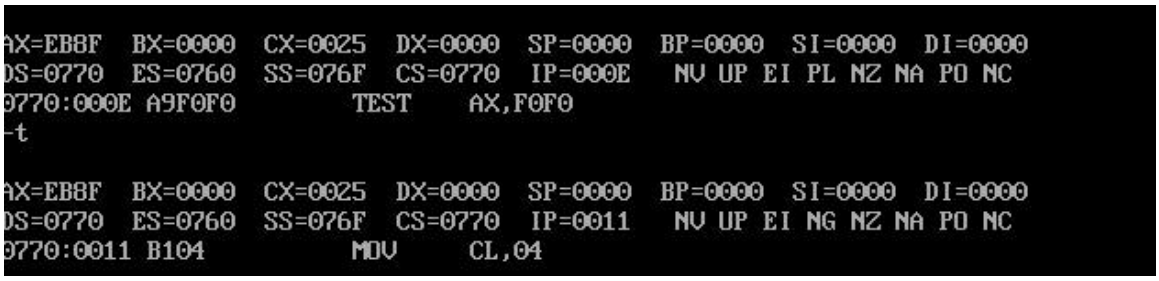
MOV AX,1470H：标志寄存器 FLAG 情况：此指令不影响标志位，各个标志位保持原值。

AND AX, AX：ZF（零标志）：结果不为 0，所以 ZF = 0。SF（符号标志）：1470H 的最高位为 0，所以 SF = 0。PF（奇偶标志）：1470H 转换为二进制后 1 的个数为奇数，所以 PF = 0。

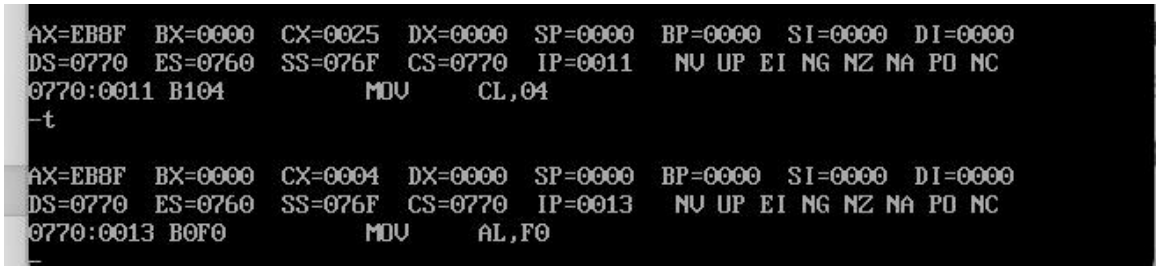
OR AX, AX：与 AND AX, AX 类似。ZF = 0。SF = 0。PF = 0。

NOT AX：CF 和 OF 不受影响。SF：EBAFH 的最高位为 1，所以 SF = 1。ZF：结果不为 0，所以 ZF = 0。PF：EBAFH 转换为二进制后 1 的个数为奇数，所以 PF = 0。

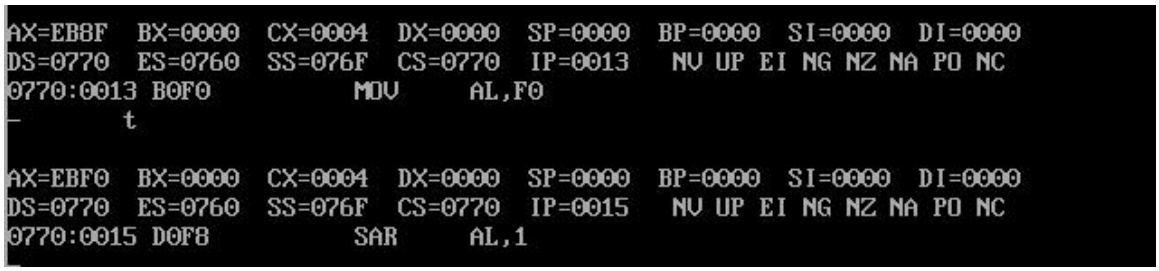
TEST AX,0F0F0H ：ZF：结果不为 0，所以 ZF = 0。SF：结果的最高位为 0，所以 SF = 0。PF：根据结果中 1 的个数的奇偶性设置，这里结果中 1 的个数为偶数，所以 PF = 1。



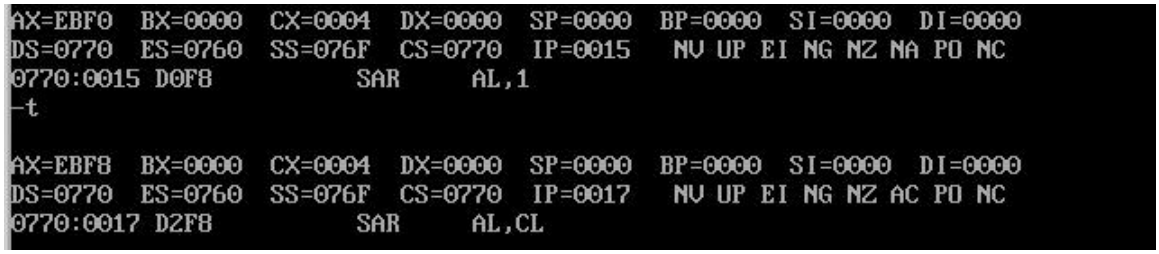
MOV CL,4：无影响



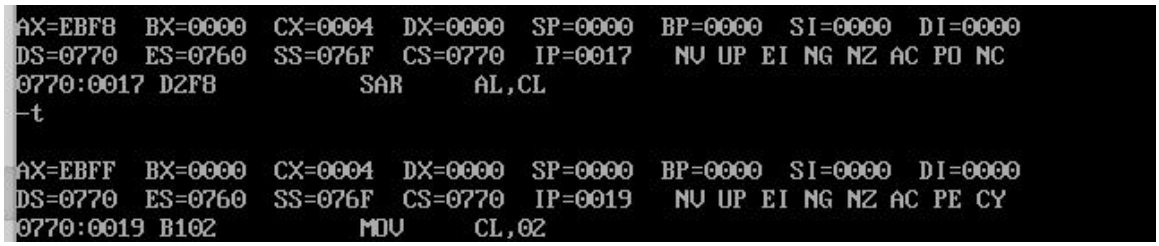
MOV AL, 0F0H：无影响



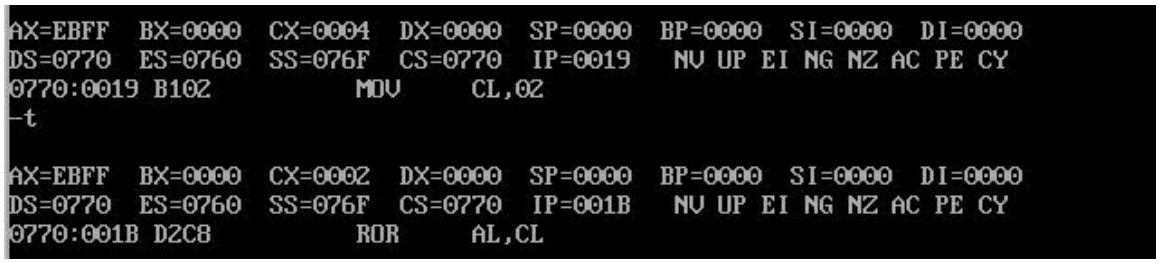
SAR AL,1：SF：78H 的最高位为 0，所以 SF = 0。ZF：结果不为 0，所以 ZF = 0。PF：78H 转换为二进制后 1 的个数为偶数，所以 PF = 1。



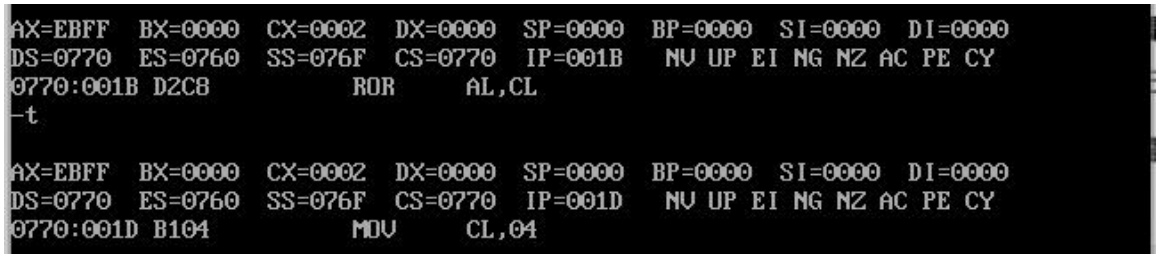
SAR AL, CL 标志寄存器 FLAG 情况：SF：0FH 的最高位为 0，所以 SF = 0。ZF：结果不为 0，所以 ZF = 0。PF：0FH 转换为二进制后 1 的个数为奇数，所以 PF = 0。



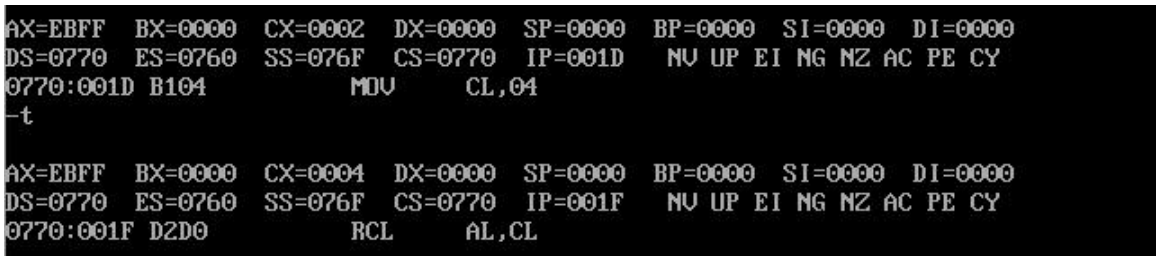
MOV CL,2:无影响



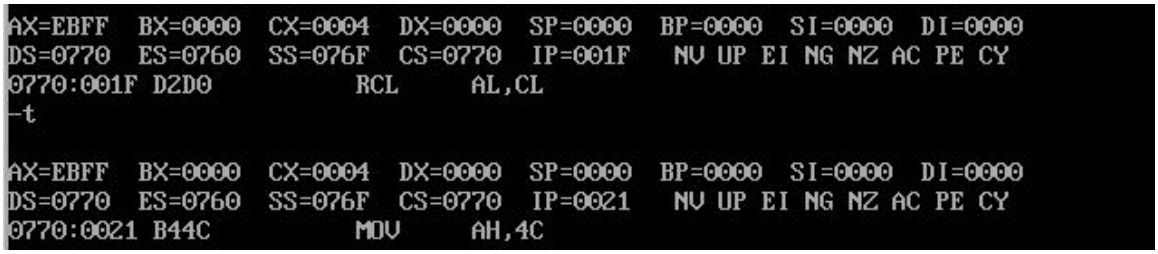
ROR AL, CL：CF：根据循环右移时移出的位设置，这里移出的位为 0 和 1，所以 CF = 0。SF：C3H 的最高位为 1，所以 SF = 1。ZF：结果不为 0，所以 ZF = 0。PF：C3H 转换为二进制后 1 的个数为奇数，所以 PF = 0。



MOV CL,4：无影响



RCL AL,CL：CF：循环左移时，原来的最高位进入 CF，这里原来的最高位为 0，所以 CF = 0。SF：0F00H 的最高位为 1，所以 SF = 1。ZF：结果不为 0，所以 ZF = 0。PF：0F00H 转换为二进制后 1 的个数为偶数，所以 PF = 1。



实验三：

（1）

```
DATAS SEGMENT
;此处输入数据段代码
X DW 200h ; X 初始值为 200
Y DW 300h ; Y 初始值为 300
Z DW 400h ; Z 初始值为 400
V DW 500h ; V 初始值为 500

DATAS ENDS
```

2）

程序首先将数据段地址存入 AX，再将其赋给数据段寄存器 DS，确保程序能正确访问数据段中的变量。

接着，将变量 X 的值加载到 AX，将变量 Y 的值加载到 BX，然后执行有符号乘法 IMUL BX，将结果的低 16 位存入 CX，高 16 位存入 BX。

然后将变量 Z 的值加载到 AX，执行 CWD 将 AX 扩展为双字，再与 CX 相加，高位的进位加到 BX。

接着将 CX 减去十六进制数 0540H，并考虑借位对 BX 进行减法操作。

再将变量 V 的值加载到 AX，执行 CWD 扩展后，减去 CX，高位的借位减去 BX。

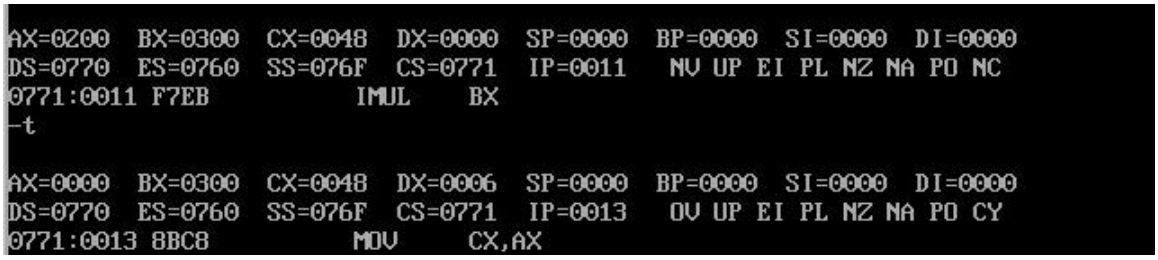
最后将变量 X 的值加载到 BX，执行有符号除法 IDIV BX。

程序最后通过中断退出。

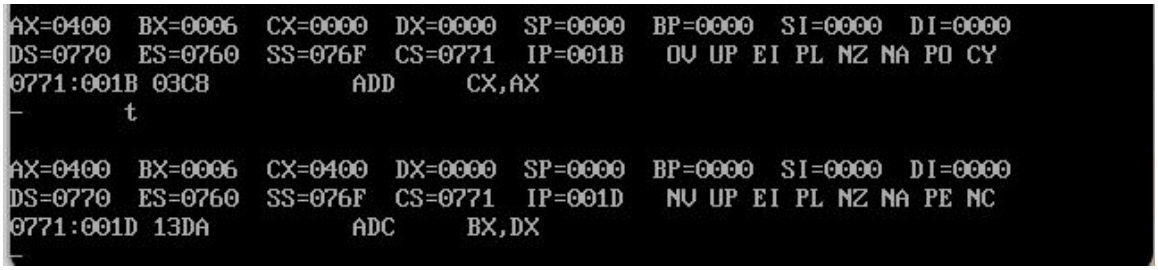
（3）第一次 CWD 在将变量 Z 的值加载到 AX 后执行，其作用是将 AX 中的有符号数扩展为双字，为后续与 CX 和 BX 进行加法运算做准备，确保能正确处理有符号数的加法操作，尤其是当结果可能超出 16 位范围时。第二次 CWD 在将变量 V 的值加载

到 AX 后执行，同样是将 AX 中的有符号数扩展为双字，为后续与 CX 和 BX 进行减法运算做准备，确保在有符号数减法操作中能正确处理可能出现的借位情况。
（4）最后的结果存放位置及结果分析：对于有符号除法 IDIV BX，商存放在 AX 中，余数存放在 DX 中。

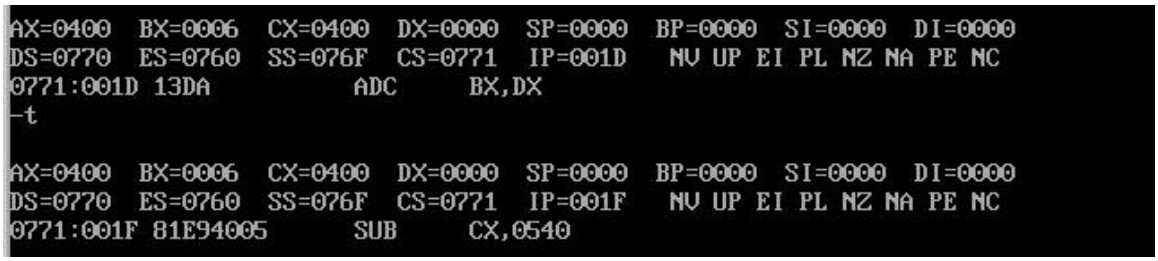
IMUL BX：AX 为 200h，BX 为 300h，有符号乘法结果为 60000h。高位存入 DX，低位存入了 AX。



ADD CX, AX：将 CX 与扩展后的 AX 相加。结果：CX = 0000h + 400h = 400h。



ADC BX,DX：带进位加法，将 BX 与扩展后的 DX 以及上一步的进位相加。结果：BX = 6h + 0000h = 6h。



SUB CX,0540H：将 CX 减去十六进制数 0540H。结果：CX = 400h - 0540h = FAC0。

```
AX=0400  BX=0006  CX=0400  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0771  IP=001F   NV UP EI PL NZ NA PE NC
0771:001F 81E94005      SUB     CX,0540
-t

AX=0400  BX=0006  CX=FEC0  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0771  IP=0023   NV UP EI NG NZ NA PE CY
0771:0023 83DB00        SBB     BX,+00
```

SUB AX,CX：将扩展后的 AX 减去 CX。结果：AX = 500h - FACH = 500h + 0540h - 400h = 640h。

```
AX=0500  BX=0005  CX=FEC0  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0771  IP=002A   NV UP EI PL NZ NA PE NC
0771:002A 2BC1          SUB     AX,CX
-t

AX=0640  BX=0005  CX=FEC0  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0771  IP=002C   NV UP EI PL NZ NA PO CY
0771:002C 1BD3          SBB     DX,BX
```

IDIV BX：用前面结果除以 X（200h），商存入 AX，余数存入 DX。

最后结果商为 FD04h，余数为 FE40h

```
AX=0640  BX=0200  CX=FEC0  DX=FFFA  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0771  IP=0032   NV UP EI NG NZ AC PE CY
0771:0032 F7FB          IDIV    BX
-t

AX=FD04  BX=0200  CX=FEC0  DX=FE40  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0771  IP=0034   NV UP EI NG NZ AC PE CY
0771:0034 B44C          MOV     AH,4C
-t
```

实验四：

（1）可以用 xor dx,dx 来 代替

（2）将 NUMW 设为 2000H，bx 设为 20H。结果应该为商为 100，余数为 0。一致。

```
AX=0100  BX=0020  CX=009B  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0778  IP=0017   NV UP EI PL NZ NA PO NC
0778:0017 B44C          MOV     AH,4C
-d 0000
0770:0000  00 20 00 01 00 00 00 00-00 00 00 00 00 00 00 00   . ..............
```

（3）将无符号除法指令 div bx 改为有符号除法指令 idiv bx。需要注意的是，此时被除数应该存放在 dx:ax 中，如果原来的被除数是 16 位的，需要先将其扩展到 32 位，使用 cwd 指令（将 ax 中的有符号数扩展为双字，即 dx:ax）。

（4）取正数时，NUMW 设为 1000H（正数），bx 设为 10H（正数）。取负数时，NUMW 设为 -1000H（假设以补码形式存储为 F000H），bx 设为 -10H（假设以补码形式存储为 FFF6H）。

对于正数情况，1000H 除以 10H，商为 100H，余数为 0。执行有符号除法 idiv bx 后，结果与无符号除法相同。

```
AX=1000  BX=0010  CX=009B  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0778  IP=000E   NV UP EI PL NZ NA PO NC
0778:000E F7F3          DIV     BX
-t

AX=0100  BX=0010  CX=009B  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0778  IP=0010   NV UP EI PL NZ NA PO NC
0778:0010 A30200        MOV     [0002],AX                       DS:0002=0000
```

对于负数情况，F000H 表示 - 4096，FFF6H 表示 - 10（补码计算），-4096 除以 -10，商为 409（余数为 -6，但在有符号除法中余数与被除数同号，所以余数为 4090）。

```
AX=F000  BX=FFF6  CX=009B  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0778  IP=000E   NV UP EI PL NZ NA PO NC
0778:000E F7F3          DIV     BX
-t

AX=0000  BX=FFF6  CX=009B  DX=F000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0778  IP=0010   NV UP EI PL NZ NA PO NC
0778:0010 A30200        MOV     [0002],AX                       DS:0002=0000
```

```
AX=F000  BX=FFF6  CX=009B  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0778  IP=000E   NV UP EI PL NZ NA PO NC
0778:000E F7FB          IDIV    BX
-t

AX=E800  BX=FFF6  CX=009B  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0778  IP=0010   NV UP EI PL NZ NA PO NC
0778:0010 A30200        MOV     [0002],AX                       DS:0002=0000
```

结论：对于有符号数的除法，需要使用有符号除法指令 idiv。结果会根据操作数的符号进行正确的计算，并且余数的符号与被除数相同。

（5）将 bx 中的除数改为 0 的结果：出现 div error。

实验五：

（1）该代码段实现的功能：该代码段实现了计算 1*2 + 2*3 + 3*4 +... + n*(n+1)的值，直到 n*(n+1)等于或大于 002AH（42）为止。
（2）每次循环中寄存器的含义：AX：存放每次循环中 AL（当前的 BL 值）和 BL（下一个数）的乘积，即 n*(n + 1)。CX：累加每次循环中 n*(n + 1)的值，最终存放表达式的总和。
（3）循环体执行次数及如何改变循环次数：直到某一次循环使得 AX >= 002AH 时循环结束。若要改变循环次数，可以修改比较指令中的比较值 002AH，或者修改初始值 BL 的初始值以及每次循环中 BL 的增量，这些都会影响循环的结束条件和循环次数。还可以修改循环体内部的运算逻辑，例如改变乘法运算的因子或者加法运算的被加数等，也会间接影响循环次数。

5 实验分析与总结

（1）继续运用 debug 来调试程序。

（2）学习一些算术指令

（3）学会阅读更复杂的程序