

循环程序设计

一、实验目的：

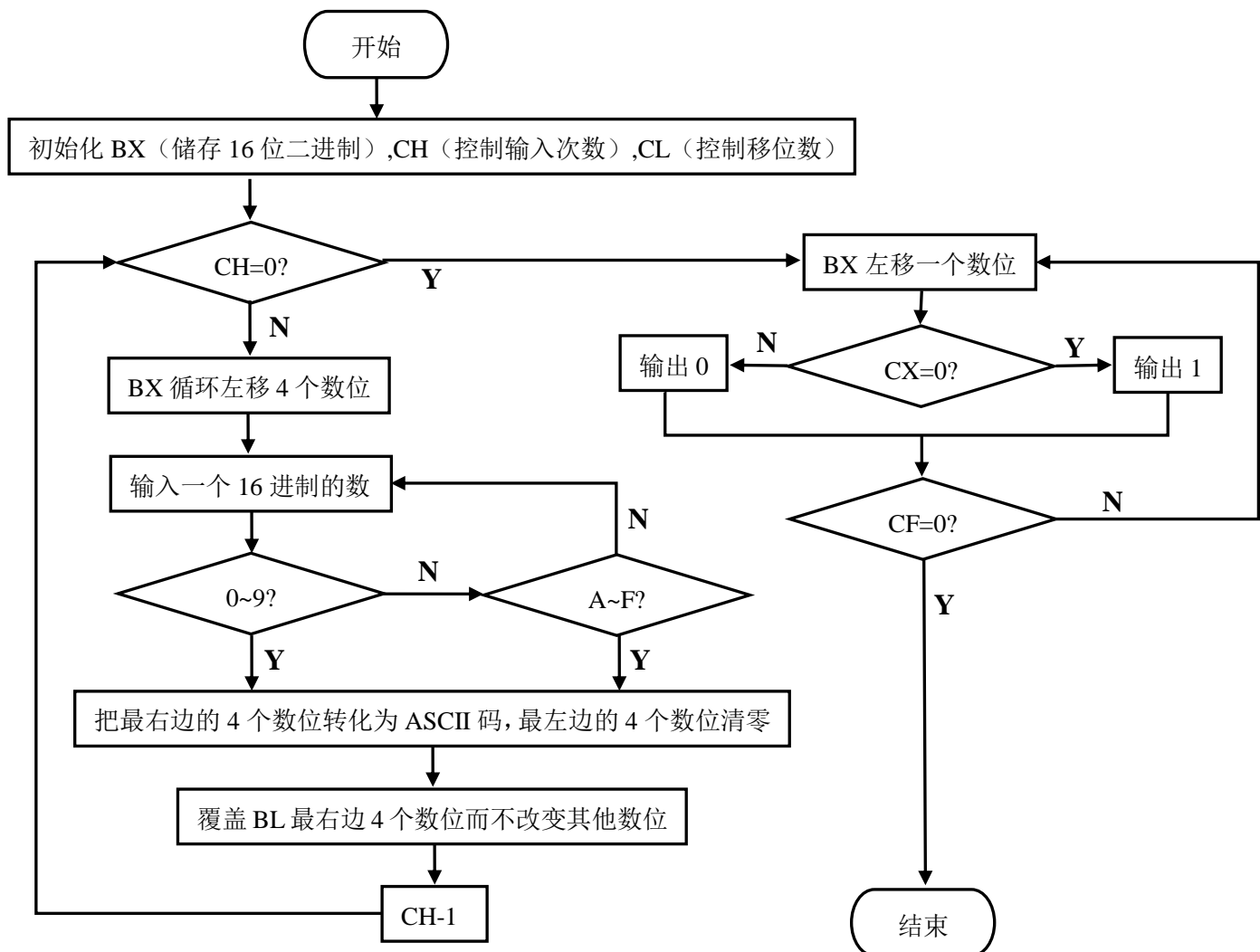
1. 加深对循环程序的理解。
2. 能构造出正确的循环结构并能实现较复杂的算法

二、实验内容：

1. 编制程序，要求如下：从键盘接收一个四位的 16 进制数，在终端上显示与它等值的二进制数。编辑、汇编、连接这个汇编语言源程序，形成 .EXE 文件。

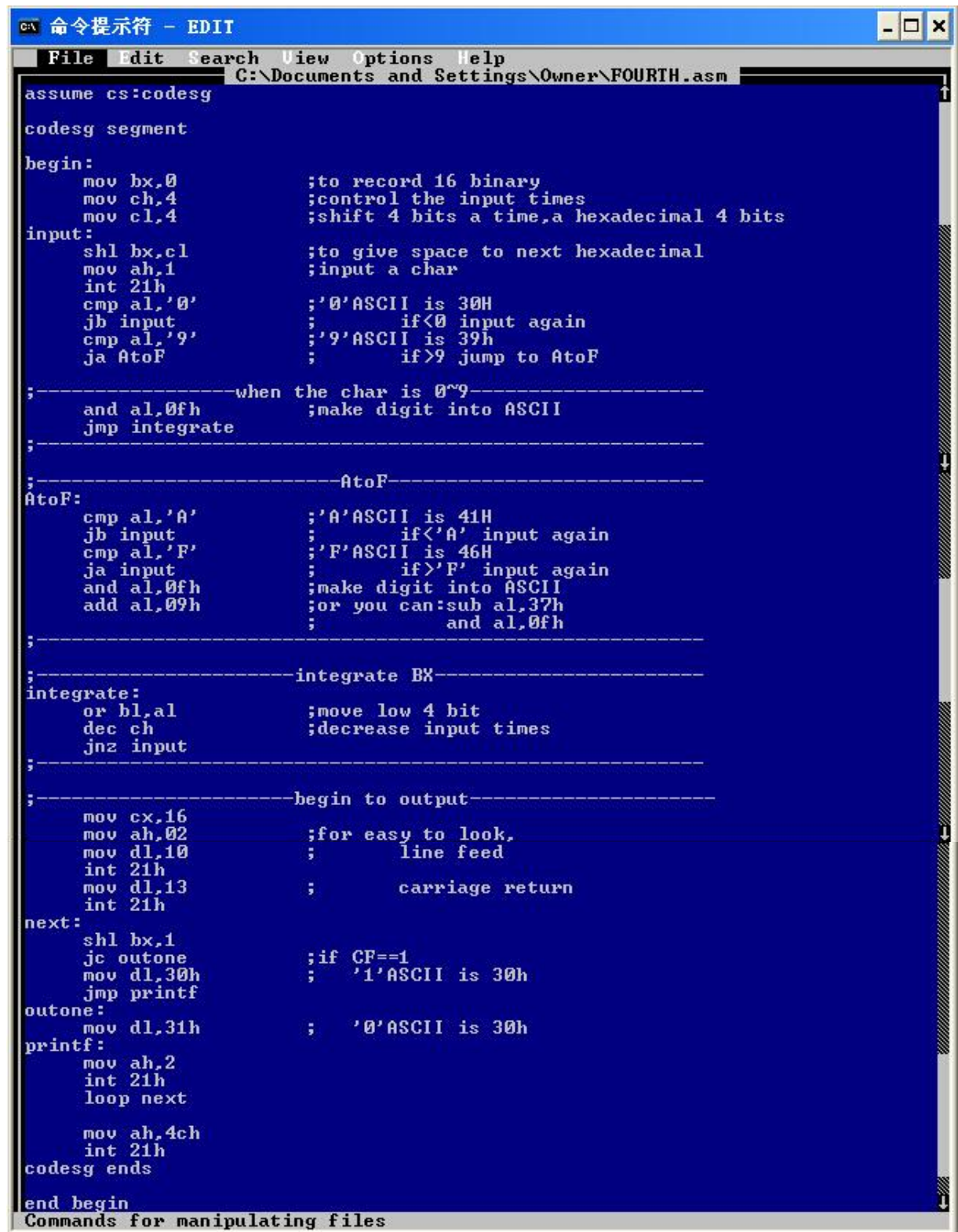
三、实验主要步骤：

1. 根据要求，画出程序框图，写出汇编源代码。



2.上机编辑源程序，并汇编、连接、调试运行，查看结果。

(1) 源代码:



```
命令提示符 - EDIT
File Edit Search View Options Help
C:\Documents and Settings\Owner\FOURTH.asm

assume cs:codesg
codesg segment
begin:
    mov bx,0           ;to record 16 binary
    mov ch,4           ;control the input times
    mov cl,4           ;shift 4 bits a time,a hexadecimal 4 bits
input:
    shl bx,cl          ;to give space to next hexadecimal
    mov ah,1           ;input a char
    int 21h
    cmp al,'0'         ;'0'ASCII is 30H
    jb input           ;if<0 input again
    cmp al,'9'         ;'9'ASCII is 39h
    ja AtoF            ;if>9 jump to AtoF

;-----when the char is 0~9-----
    and al,0fh         ;make digit into ASCII
    jmp integrate
;-----

;-----AtoF-----
AtoF:
    cmp al,'A'         ;'A'ASCII is 41H
    jb input           ;if<'A' input again
    cmp al,'F'         ;'F'ASCII is 46H
    ja input           ;if>'F' input again
    and al,0fh         ;make digit into ASCII
    add al,09h         ;or you can:sub al,37h
                        ;and al,0fh
;-----

;-----integrate BX-----
integrate:
    or bl,al           ;move low 4 bit
    dec ch             ;decrease input times
    jnz input
;-----

;-----begin to output-----
    mov cx,16
    mov ah,02          ;for easy to look,
    mov dl,10          ;line feed
    int 21h
    mov dl,13          ;carriage return
    int 21h
next:
    shl bx,1
    jc outone          ;if CF==1
    mov dl,30h         ;'1'ASCII is 30h
    jmp printf
outone:
    mov dl,31h         ;'0'ASCII is 30h
printf:
    mov ah,2
    int 21h
    loop next

    mov ah,4ch
    int 21h
codesg ends
end begin
Commands for manipulating files
```

(2) 汇编、连接、运行，查看结果

```
C:\DOCUME~1\Owner>MASM
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988. All rights reserved.

Source filename [.ASM]: FOURTH
Object filename [FOURTH.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:

    49320 + 447205 Bytes symbol space free

    0 Warning Errors
    0 Severe Errors

C:\DOCUME~1\Owner>LINK

Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.

Object Modules [.OBJ]: FOURTH
Run File [FOURTH.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment

C:\DOCUME~1\Owner>FOURTH
FFFF
1111111111111111
C:\DOCUME~1\Owner>FOURTH
0000
0000000000000000
C:\DOCUME~1\Owner>FOURTH
ABCD
1010101111001101
C:\DOCUME~1\Owner>FOURTH
1234
0001001000110100
C:\DOCUME~1\Owner>FOURTH
EF34
1110111100110100
C:\DOCUME~1\Owner>_
```

测试结果如下：

FFFF(h)=1111111111111111(b)

0000(h)=0000000000000000(b)

ABCD(h)=1010101111001101(b)

1234(h)=0001001000110100(b)

EF34(h)=1110111100110100(b)

经检验结果均正确。

若输入错误的字符，则会忽略错误字符。

AaBCD(h)=ABCD(h)=1010101111001101(b)

结果正确。

```
C:\DOCUME~1\Owner>FOURTH
AaBCD
0000101111001101
C:\DOCUME~1\Owner>_
```

(3) Debug 模式下对程序进行调试：分别输入 A、5 和 a，查看主要寄存器的内容：

键入 A

```
AX=0161 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0009 NU UP EI PL ZR NA PE NC
0E80:0009 B401          MOV     AH,01
-T
AX=0161 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=000B NU UP EI PL ZR NA PE NC
0E80:000B CD21          INT     21 ;Read Keyboard and Echo
-T
AAX=0141 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=000D NU UP EI PL ZR NA PE NC
0E80:000D 3C30          CMP     AL,30 ;'0'
```

‘A’与字符‘0’和‘9’相比（ASCII 值相比），不属于该范围，所以跳转。

```
AAX=0141 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=000D NU UP EI PL ZR NA PE NC
0E80:000D 3C30          CMP     AL,30 ;'0'
-T
AX=0141 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=000F NU UP EI PL NZ NA PE NC
0E80:000F 72F6          JB      0007
-T
AX=0141 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0011 NU UP EI PL NZ NA PE NC
0E80:0011 3C39          CMP     AL,39 ;'9'
-T
AX=0141 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0013 NU UP EI PL NZ AC PO NC
0E80:0013 7705          JA      001A
-T
AX=0141 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=001A NU UP EI PL NZ AC PO NC
0E80:001A 3C41          CMP     AL,41 ;'A'
```

‘A’与字符‘A’和‘F’相比（ASCII 值相比），属于该范围，所以不跳转。

```
AX=0141 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=001A NU UP EI PL NZ AC PO NC
0E80:001A 3C41          CMP     AL,41 ;'A'
-T
AX=0141 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=001C NU UP EI PL ZR NA PE NC
0E80:001C 72E9          JB      0007
-T
AX=0141 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=001E NU UP EI PL ZR NA PE NC
0E80:001E 3C46          CMP     AL,46 ;'F'
-T
AX=0141 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0020 NU UP EI NG NZ AC PO CY
0E80:0020 77E5          JA      0007
-T
AX=0141 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0022 NU UP EI NG NZ AC PO CY
0E80:0022 240F          AND     AL,0F
```

经过“处理”后，A 在 BX 中的最低四位

```
AX=0141 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0022 NU UP EI NG NZ AC PO CY
0E80:0022 240F          AND     AL,0F
-T
AX=0101 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0024 NU UP EI PL NZ NA PO NC
0E80:0024 0409          ADD     AL,09
-T
AX=010A BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0026 NU UP EI PL NZ NA PE NC
0E80:0026 0AD8          OR      BL,AL
-T
AX=010A BX=000A CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0028 NU UP EI PL NZ NA PE NC
0E80:0028 FECB          DEC     CH
```

A 左移 4 位，空出最低 4 位给下一个数字。

```
AX=010A BX=000A CX=0304 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0007  NU UP EI PL NZ NA PE NC
0E80:0007 D3E3          SHL     BX,CL
-T
AX=010A BX=00A0 CX=0304 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0009  NU UP EI PL NZ NA PE NC
0E80:0009 B401          MOV     AH,01
```

键入数字 5

```
AX=010A BX=00A0 CX=0304 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0009  NU UP EI PL NZ NA PE NC
0E80:0009 B401          MOV     AH,01
-T
AX=010A BX=00A0 CX=0304 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=000B  NU UP EI PL NZ NA PE NC
0E80:000B CD21          INT     21 ;Read Keyboard and Echo
-T
5AX=0135 BX=00A0 CX=0304 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=000D  NU UP EI PL NZ NA PE NC
0E80:000D 3C30          CMP     AL,30 ;'0'
```

‘5’与字符‘0’和‘9’相比（ASCII 值相比），属于该范围，所以不跳转。

```
5AX=0135 BX=00A0 CX=0304 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=000D  NU UP EI PL NZ NA PE NC
0E80:000D 3C30          CMP     AL,30 ;'0'
-T
AX=0135 BX=00A0 CX=0304 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=000F  NU UP EI PL NZ NA PE NC
0E80:000F 72F6          JB      0007
-T
AX=0135 BX=00A0 CX=0304 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0011  NU UP EI PL NZ NA PE NC
0E80:0011 3C39          CMP     AL,39 ;'9'
-T
AX=0135 BX=00A0 CX=0304 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0013  NU UP EI NG NZ AC PE CY
0E80:0013 7705          JA      001A
-T
AX=0135 BX=00A0 CX=0304 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0015  NU UP EI NG NZ AC PE CY
0E80:0015 240F          AND     AL,0F
```

经过“处理”后，5 在 BX 中的最低四位

```
AX=0141 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0022  NU UP EI NG NZ AC PO CY
0E80:0022 240F          AND     AL,0F
-T
AX=0101 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0024  NU UP EI PL NZ NA PO NC
0E80:0024 0409          ADD     AL,09
-T
AX=010A BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0026  NU UP EI PL NZ NA PE NC
0E80:0026 0AD8          OR      BL,AL
-T
AX=010A BX=000A CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0028  NU UP EI PL NZ NA PE NC
0E80:0028 FEC0          DEC     CH
```

键入小写 a

```

AX=0000 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0009 NU UP EI PL ZR NA PE NC
0E80:0009 B401          MOV     AH,01
-t
AX=0100 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=000B NU UP EI PL ZR NA PE NC
0E80:000B CD21          INT     21 ;Read Keyboard and Echo
-t
aAX=0161 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=000D NU UP EI PL ZR NA PE NC
0E80:000D 3C30          CMP     AL,30 ;'0'

```

‘a’与字符‘0’和‘9’相比（ASCII 值相比），不属于该范围，所以跳转。

```

aAX=0161 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=000D NU UP EI PL ZR NA PE NC
0E80:000D 3C30          CMP     AL,30 ;'0'
-t
AX=0161 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=000F NU UP EI PL NZ NA PO NC
0E80:000F 72F6          JB      0007
-t
AX=0161 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0011 NU UP EI PL NZ NA PO NC
0E80:0011 3C39          CMP     AL,39 ;'9'
-t
AX=0161 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0013 NU UP EI PL NZ AC PE NC
0E80:0013 7705          JA      001A
-t
AX=0161 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=001A NU UP EI PL NZ AC PE NC
0E80:001A 3C41          CMP     AL,41 ;'A'

```

‘a’与字符‘A’和‘F’相比（ASCII 值相比），也不属于该范围，所以跳转。

```

AX=0161 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=001A NU UP EI PL NZ AC PE NC
0E80:001A 3C41          CMP     AL,41 ;'A'
-t
AX=0161 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=001C NU UP EI PL NZ NA PO NC
0E80:001C 72E9          JB      0007
-t
AX=0161 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=001E NU UP EI PL NZ NA PO NC
0E80:001E 3C46          CMP     AL,46 ;'F'
-t
AX=0161 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0020 NU UP EI PL NZ AC PE NC
0E80:0020 77E5          JA      0007
-t
AX=0161 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0007 NU UP EI PL NZ AC PE NC
0E80:0007 D3E3          SHL     BX,CL

```

因为小写 a 不属于本程序 16 进制识别符，所以重新输入。

```

AX=0161 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0007 NU UP EI PL NZ AC PE NC
0E80:0007 D3E3          SHL     BX,CL
-t
AX=0161 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=0009 NU UP EI PL ZR NA PE NC
0E80:0009 B401          MOV     AH,01
-t
AX=0161 BX=0000 CX=0404 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0E70 ES=0E70 SS=0E80 CS=0E80 IP=000B NU UP EI PL ZR NA PE NC
0E80:000B CD21          INT     21 ;Read Keyboard and Echo
-t

```

四、实验结果与分析：

通过实验，进一步熟知和掌握移位指令、跳转；如何向程序输入和向显示屏输出字符。认清了以下问题：

1、ASCII 码和机器码。由于计算机处理的信息不只是数字，有可能是字符或字符串，所以计算机要能表示字符，于是出现了 ASCII。例如：我们利用用 DOS 中断向计算机输入字符 ‘A’，那么寄存器 AL 中的内容并不是 0A，而是 41，因为字符 ‘A’ 对应的 ASCII 是 41H，但是显然我们希望最后 BX（用于存放 4 个 16 进制数）对应的内容不是 41H，而是 AH，那么我们就需要通过规律让 41H 等变成用户所需要的数。同理，我们要向屏幕输出字符 ‘1’，不能让 DL=01H，因为利用 DOS 中断向屏幕输出的 DL 里内容（计算机看成 ASCII）对应的字符，所以我们应该让 DL=31H（‘1’ 对应的 ASCII 是 31H）。

2、循环指令和跳转指令的灵活运用。当 CX 寄存器用途冲突时，要灵活改用其他方法。在本实验中，因为移位指令需要 CL 的配合，如果循环输入功能用 LOOP 指令（需要 CX 配合）完成，那么就会存在冲突，这时可以不用 LOOP 改用转移指令，就能解决这个问题。

3、涉及多重跳转的程序。最重要是保持思维的清晰，包括在哪里跳转？什么条件下跳转？跳转去哪里？可以通过画程序流程图帮助理解。

4、各种条件转移指令应该配合其英文含义帮助记忆对应的条件。