实验报告一

一、实验目的：

掌握常用的DOS目录操作类命令、文件操作类命令以及其它命令的格式及使用方法。

二、实验任务：

1．练习目录操作类命令：MD、CD、RD、DIR、PATH、TREE；

2．练习文件操作类命令：COPY、TYPE、REN、DEL；

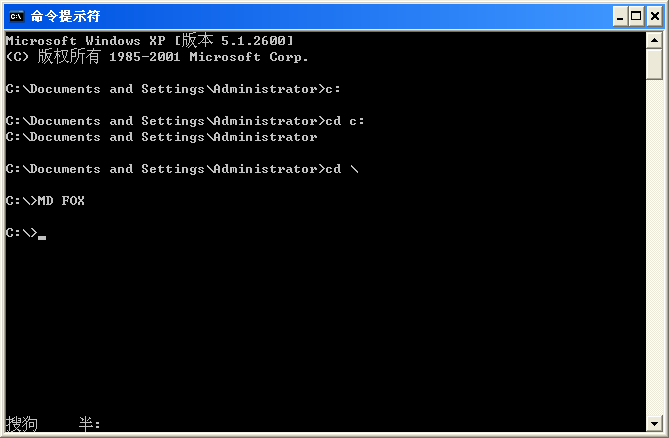
3．其它命令：CLS、VER、DATA、TIME。

二、实验过程：

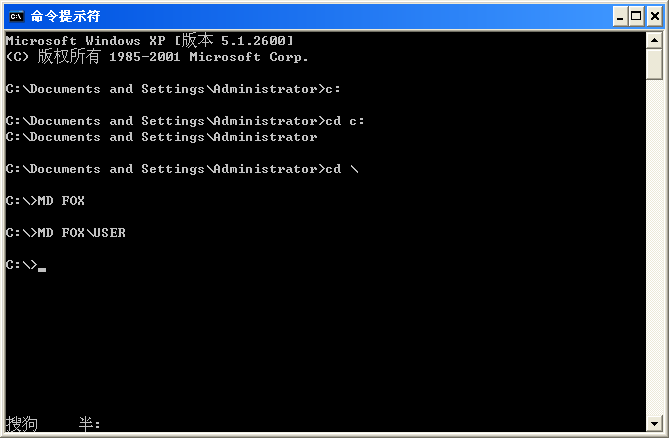
1．目录操作类命令：

（1）MD——建立子目录：

①在C盘的根目录创建名为FOX的子目录：

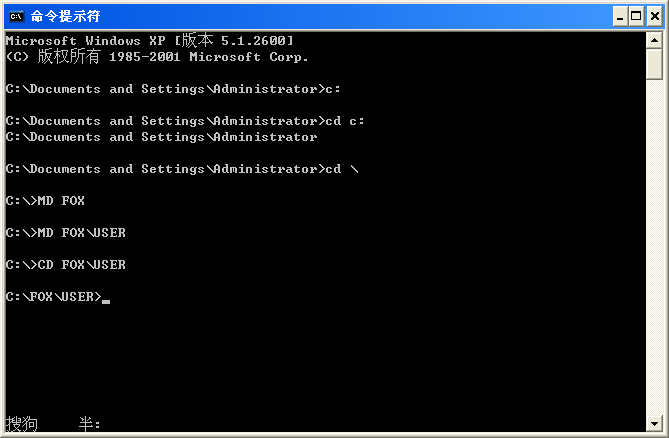


②在FOX子目录下再创建USER子目录：

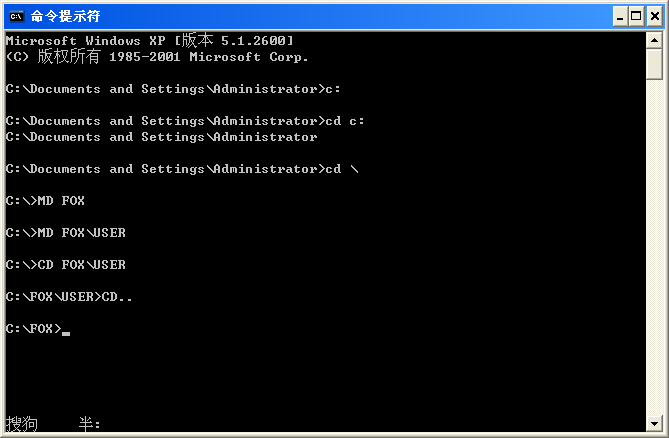


（2）CD——改变当前目录：

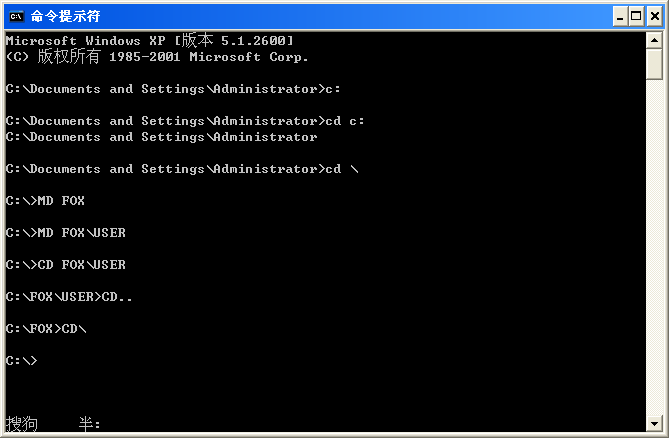
①进入到USER子目录：



②从USER子目录退回到子目录：



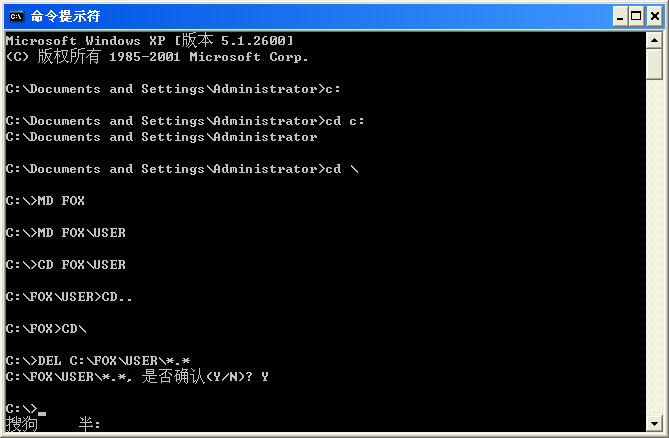
③返回到根目录：



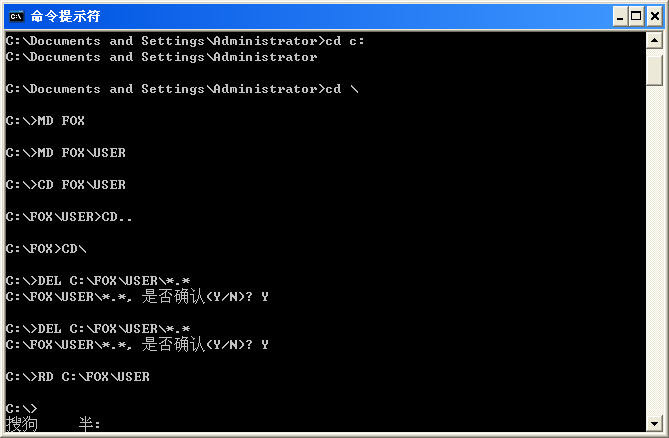
（3）RD——删除子目录：

把C盘FOX子目录下的USER子目录删除，操作如下：

①先将USER子目录下的文件删空：

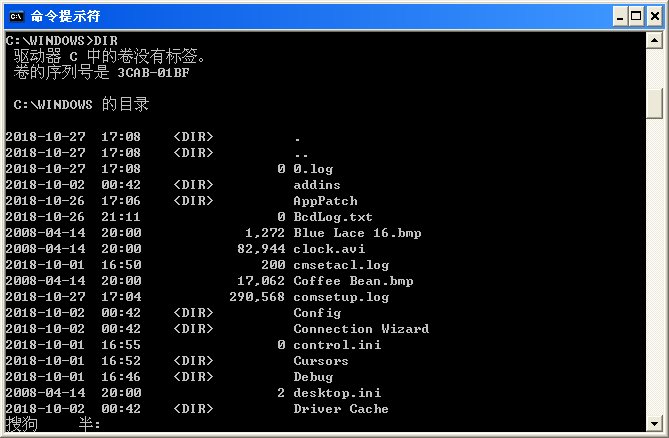


②删除USER子目录：

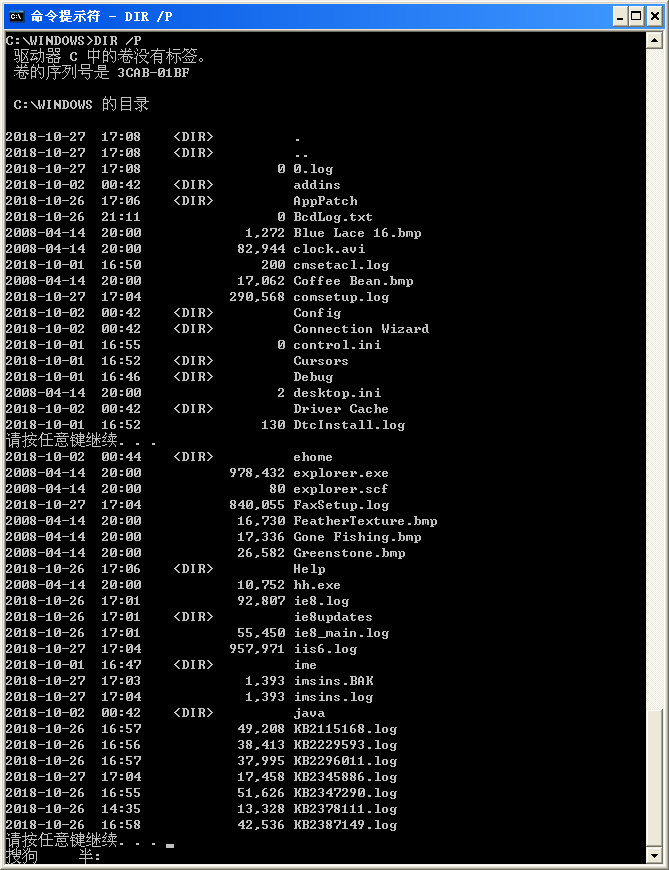


（4）DIR——显示磁盘目录：

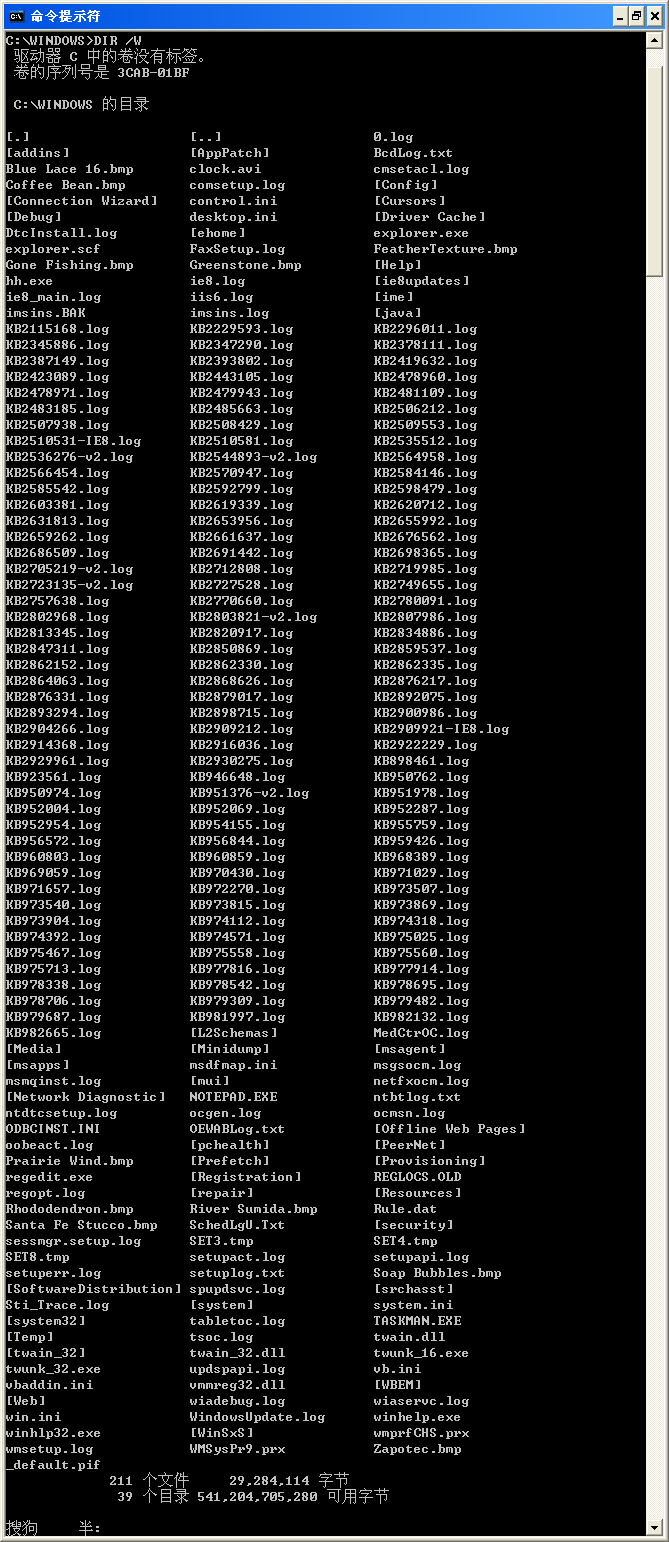
①显示C:\WINDOWS目录的内容：



②使用/P参数分面显示目录内容：

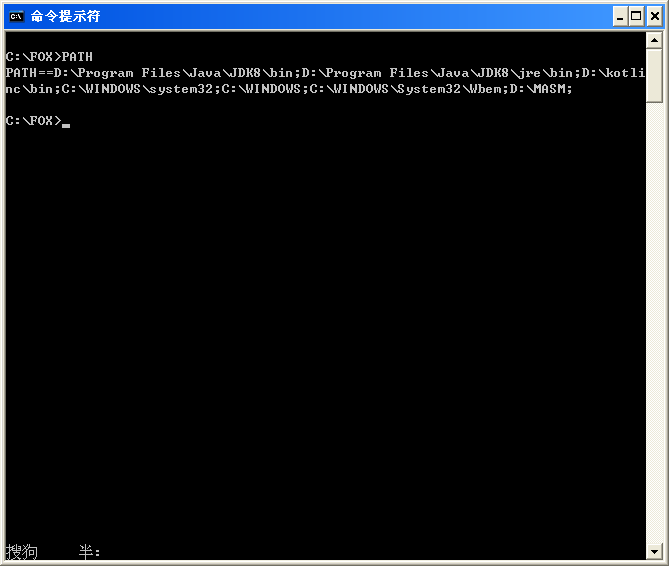


③使用/W参数只显示文件名：



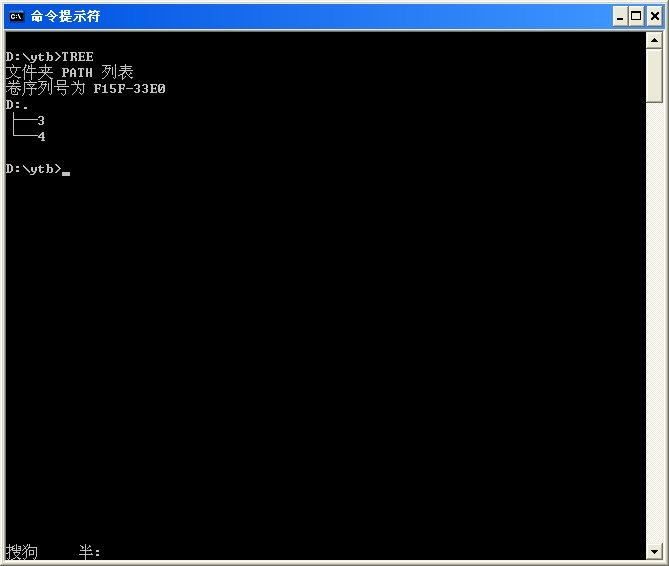
（5）PATH——设置可执行文件的搜索路径：

显示目前所设的路径：

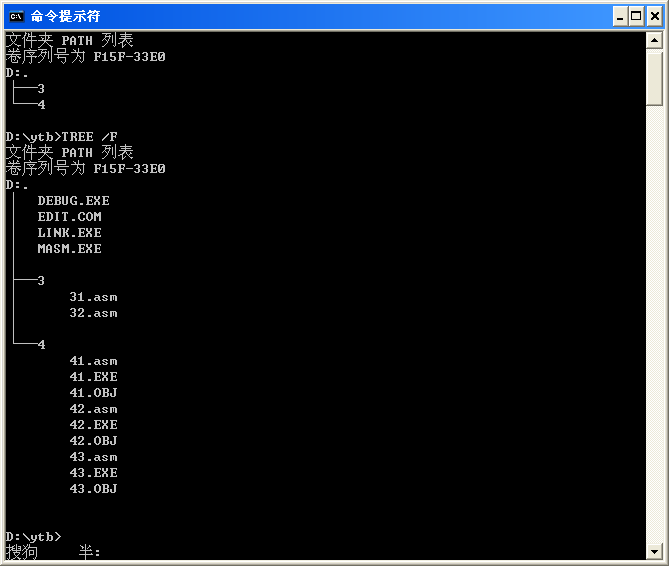


（6）TREE——显示磁盘目录结构：

①显示文件夹的目录结构：



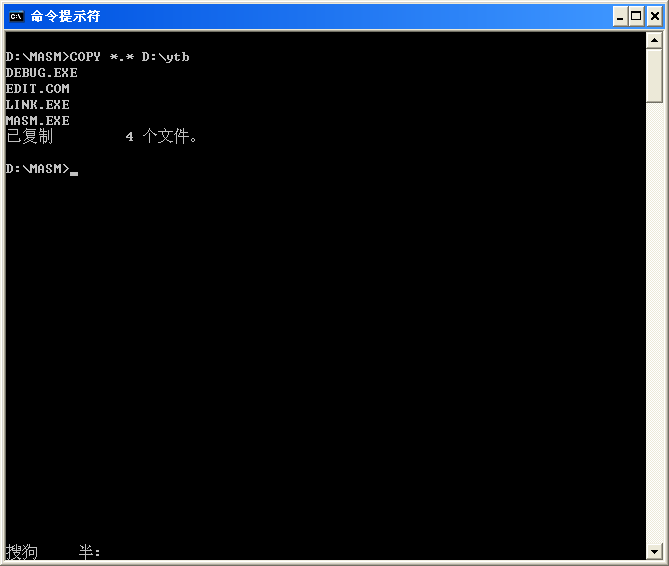
②使用/F参数，显示目录和目录下的文件：



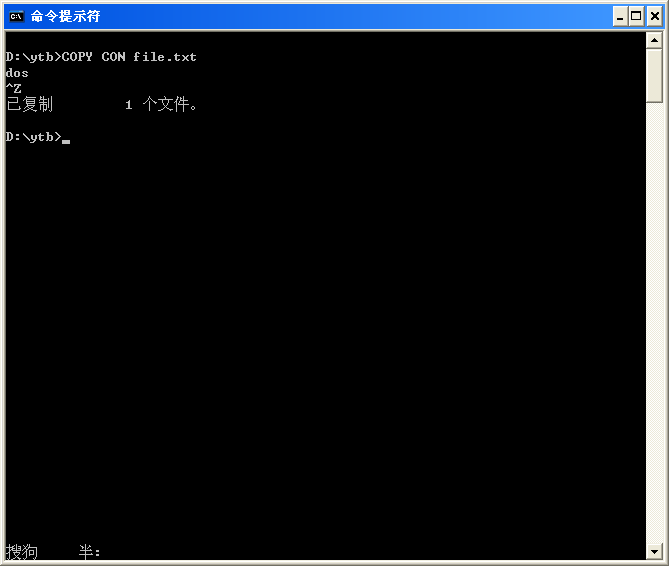
2．文件操作类命令：

（1）COPY——文件复制：

①复制D:\MASM中的所有文件到自己的文件夹：

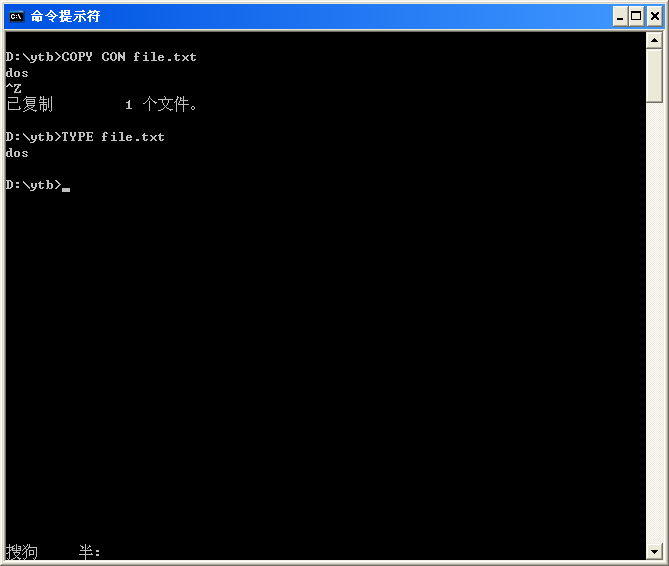


②从键盘上输入数据建立文件：

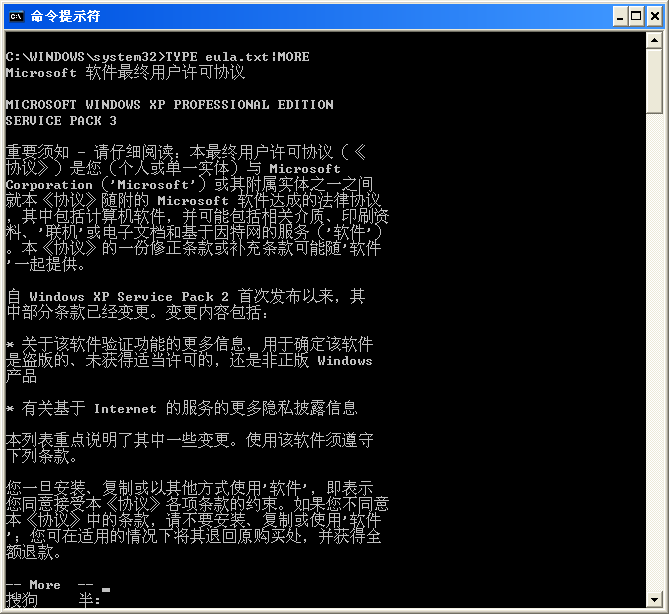


（2）TYPE——显示文件内容：

①显示file.txt的内容：

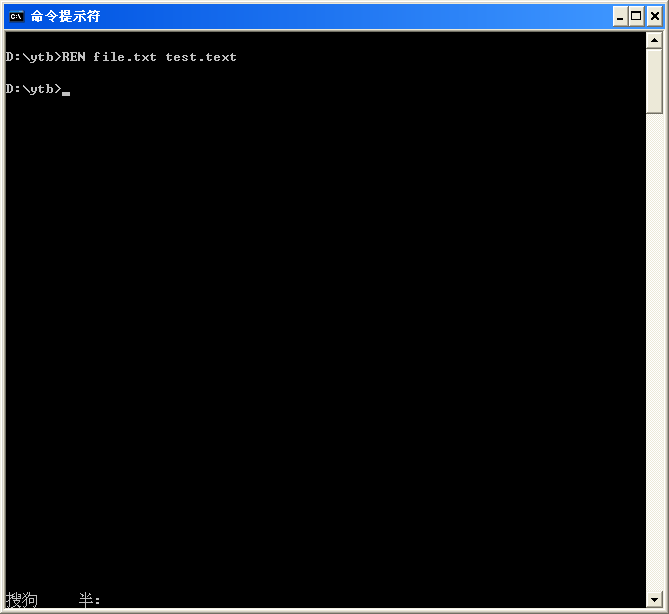


②分屏显示内容：



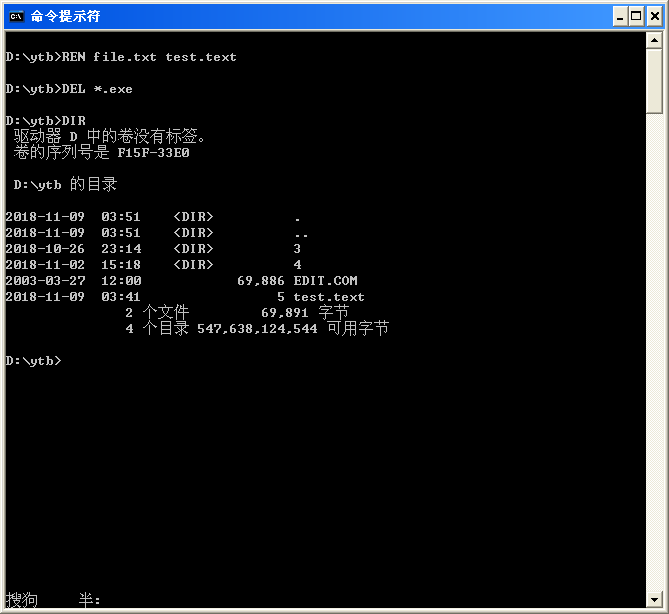
（3）REN——文件改名：

将file.txt改名为test.text：

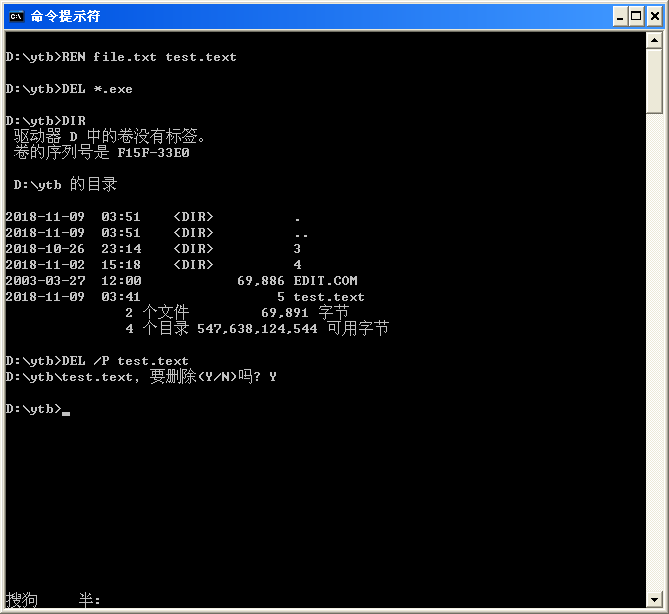


（4）DEL——删除文件：

①删除当前目录下的所有exe文件：



②使用/P参数删除前询问：



3．其它命令：

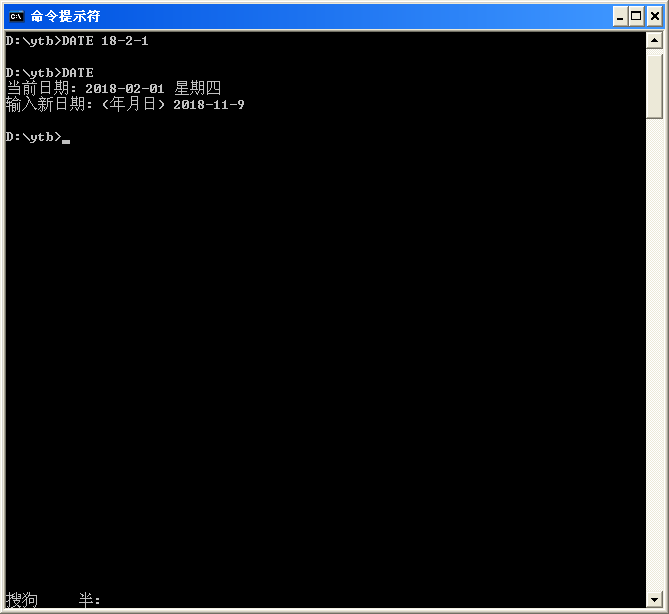
（1）CLS——清屏幕：

（2）VER——查看系统版本号：

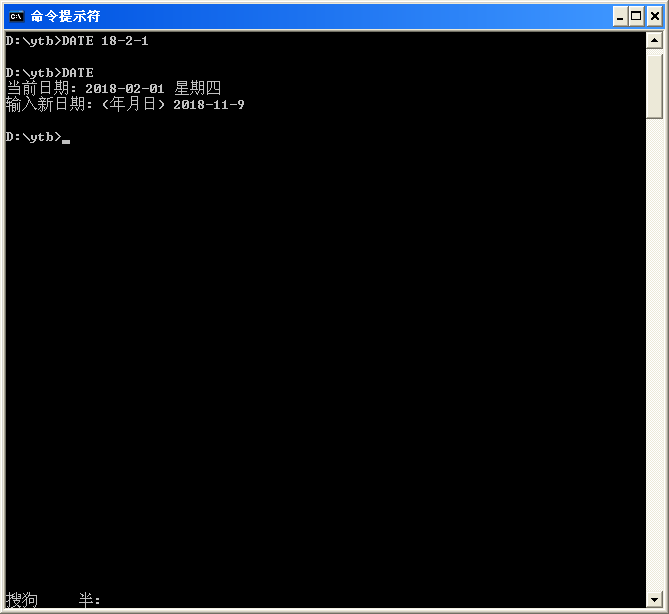


（3）DATE——日期设置：

①设置系统日期为2018年2月1日：



②显示系统时间并设置新日期：



三、实验心得：

通过本次实验，我强化了对DOS命令的理解，学习了一些常用命令（如：COPY命令从键盘输入数据建立文件）不曾使用过的用法，学习这些DOS命令，为进一步学习书写批处理程序让计算机自动解决一些问题做了铺垫，也为后续在Windows下利用DEBUG调试汇编程序奠定了基础。

实验报告二

一、实验目的：

了解什么是Debug，学习使用Debug的基本功能，熟练使用Debug调试汇编程序。

二、实验任务：

1．使用Debug，将下面的程序段写入内存，逐条执行，观察每条指令执行后CPU中相关寄存器的内容变化。

|  |  |
| --- | --- |
| 机器码 | 汇编指令 |
| b8 20 4e  05 16 14  Bb 00 20  01 d8  89 c3  01 d8  b8 1a 00  bb 26 00  00 d8  00 dc  00 c7  b4 00  00 d8  04 9c | mov ax, 4e20h ;(ax)=4e20h  add ax, 1416h ;(ax)=(ax)+1416h  mov bx, 2000h ;(bx)=2000h  add ax, bx ;(ax)=(ax)+(bx)  mov bx, ax ;(bx)=(ax)  add ax, bx ;(ax)=(ax)+(bx)  mov ax, 001ah ;(ax)=001ah  mov bx, 0026h ;(bx)=0026  add al, bl ;(al)=(al)+(bl)  add ah, bl ;(ah)=(ah)+(bl)  add bh, al ;(bh)=(bh)+(al)  mov ah, 0 ;(ah)=0  add al, bl ;(al)=(al)+(bl)  add al,9ch ;(al)=(al)+9ch |

提示，可用E命令和A命令以两种方式将指令写入内存。注意用T命令执行时，CS:IP的指向。

2．将下面3条指令写入从2000:0开始的内存单元中，利用3条指令计算2的8次方。

mov ax, 1

add ax, ax

jmp 2000: 0003

3．查看内存中的内容：

PC机主板上的ROM中写有一个生产日期，在内存FFF00H~FFFFFH的某几个单元中，请找到这个生产日期并试图改变它。

提示，如果读者对实验的结果感到疑惑，请仔细阅读第1章中的1.15节。

4．向内存从B8100h开始的单元中填入数据，如：

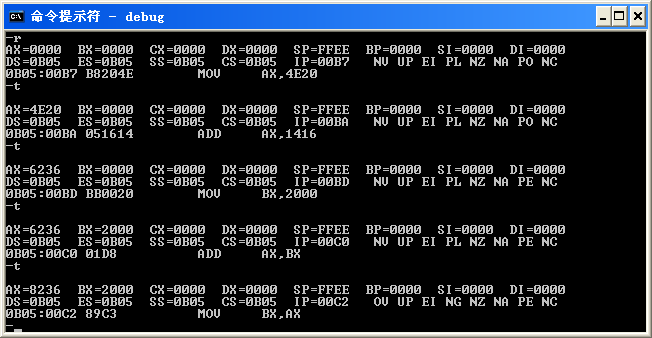
-e B810:0000 01 01 02 02 03 03 04 04

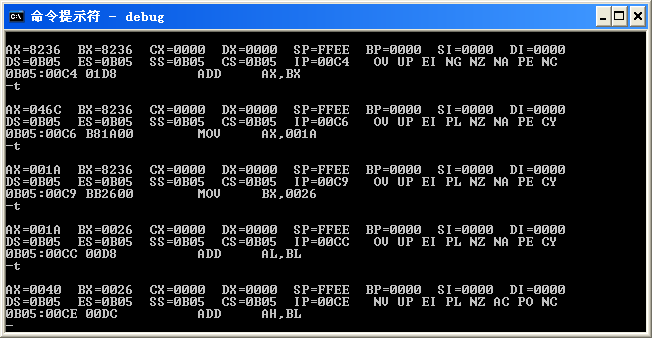
请读者先填写不同的数据，观察产生的现象；再改变填写的地址，观察产生的现象。

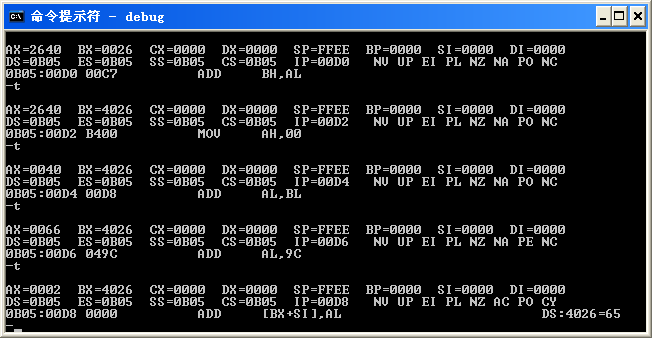
提示，如果读者对实验的结果感到疑惑，请仔细阅读第1章、中的1.15节。

三、实验过程：

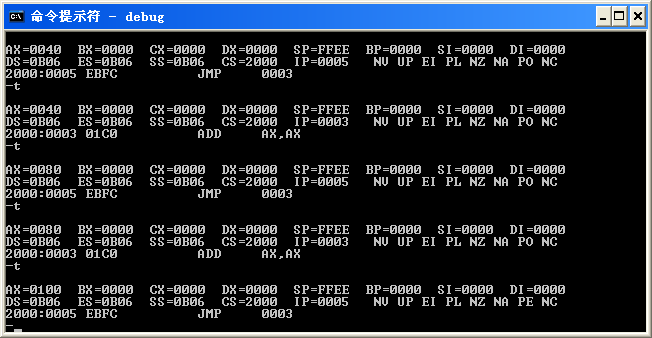
1．向内存中输入相应的指令，用t命令单步执行，记录每次CPU寄存器中的内容：



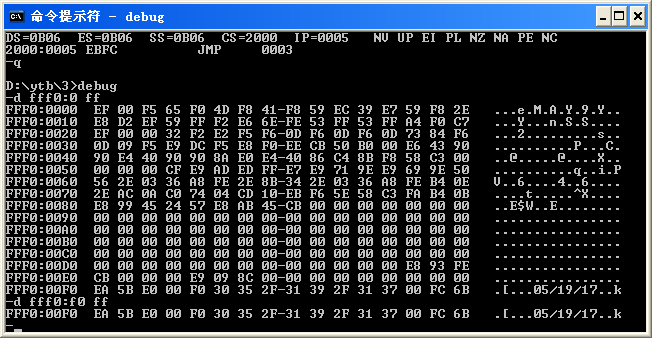




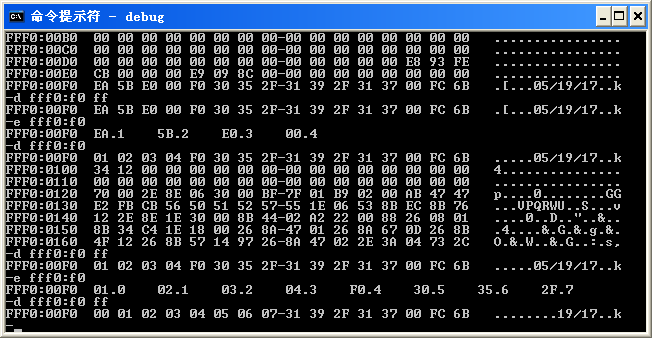
2．通过多次执行命令，每次寄存器ax的值翻倍，可以得到2的8次方为100h：



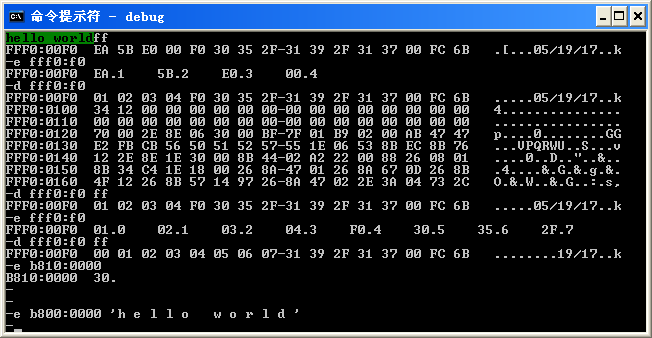
3．通过调用d命令可以看到生产日期为：2017年5月19日



通过e命令改变地址为FFF00~FFFFF的内容，d命令查看，正常情况其值不会被改变，但由于此次是在虚拟机中运行，其值有变化：



4．修改b800:0000处的内容，发现在控制台的左上角会发现彩色的字符：



四、实验总结：

在Debug中：

R命令：查看，修改CPU中寄存器的内容  
 D命令：查看内存中的内容  
 E命令：修改内存中的内容  
 U命令：将内存中的内容解释为机器指令和对应的汇编指令  
 T命令：执行CS:IP指向的内存单元处的命令  
 A命令：以汇编指令的形式向内存中写入指令

通过熟练掌握这些命令，为我们后续编程打下基础。

实验报告三

一、实验目的：

通过实验了解DOS常用命令，汇编就是把用汇编语言编写的源程序翻译（汇编）成机器语言的目标程序。汇编程序可以使用小汇编程序（ASM）也可以用宏汇编程序（MASM），用于宏汇编程序不但可以代替ASM，而且可以汇编具有宏定义的汇编程序，因此我们在汇编程序时使用宏汇编程序（MASM）。开发汇编语言源程序的主要步骤有哪些？首先用EDIT等编辑程序产生汇编语言的源程序，源程序是用汇编语言的语句编写的且不能为机器所识别的程序，所以要经过汇编程序加以翻译，因此汇编程序的作用就是把源文件转换成用二进制代码表示的目标文件（称为OBJ文件）。在转换的过程中，如果源程序中有语法错误，则汇编结束后，汇编程序将指出源程序中的错误信息，如非法格式，未定义的助记符、标号，漏掉操作数等。用户还可以用编辑程序来修改源程序中的错误，最后得到无语法错误的目标文件。目标文件虽然已经是二进制文件，但它还不能直接上机运行，必须经过连接程序（LINK）把目标文件与库文件或其他目标文件连接在一起形成可执行文件（EXE文件），才可以在机器上运行。

二、实验任务：

1．dos命令建立目录拷贝文件（edit.com、masm.exe、link.exe、debug.exe）执行宏汇编程序。

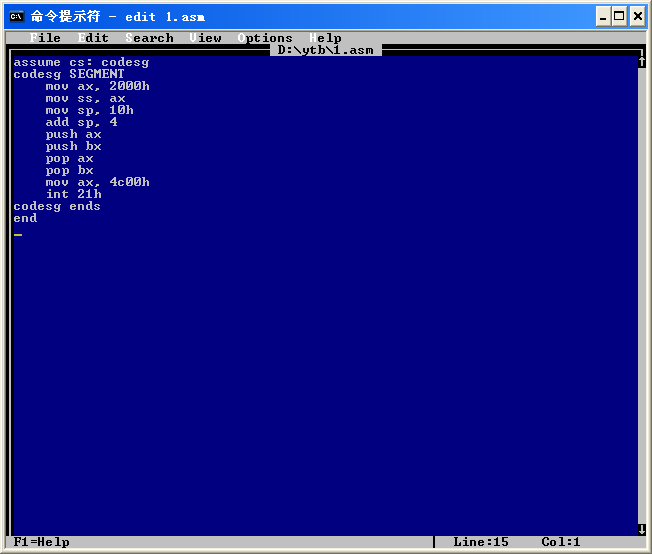
2．通过调用程序DEBUG的主要命令，熟悉各种命令的用法。

三、实验过程：

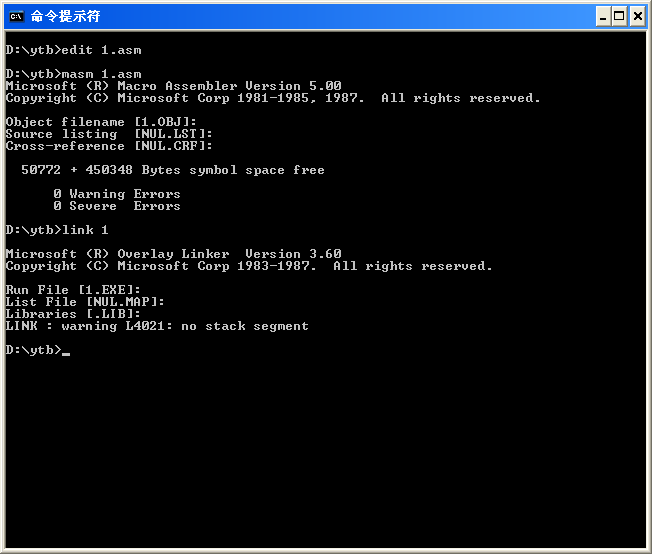
1．将下面的程序保存为1.asm，将其生成可执行文件1.exe。

|  |
| --- |
| assume cs: codesg  codesg SEGMENT  mov ax, 2000h  mov ss, ax  mov sp, 10h  add sp, 4  push ax  push bx  pop ax  pop bx  mov ax, 4c00h  int 21h  codesg ends  end |

（1）编辑1.asm：

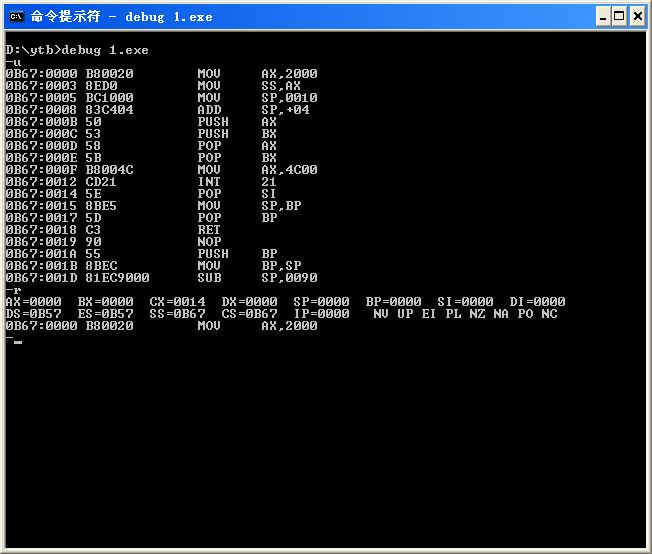


（2）使用MASM编译生成目标文件1.obj，使用LINK将目标程序连接定位，生成可执行文件1.exe：



2．使用DEBUG跟踪1.exe，写出每一步执行后，相关寄存器的内容。

（1）使用DEBUG跟踪1.exe：



（2）记录每一步执行后相关寄存器的内容

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 行号 | 机器码 | 字节数 | 汇编指令 | 汇编指令执行后寄存器的内容 | | | |
| ax | bx | ip | sp |
| 1  2  3  4  5  6  7  8  9  10 | B80020  8ED0  BC1000  83C404  50  53  58  5B  B8004C  CD21 | 3  2  3  3  1  1  1  1  3  2 | MOV AX, 2000  MOV SS, AX  MOV SP, 0010  ADD SP, +04  PUSH AX  PUSH BX  POP AX  POP BX  MOV AX, 4C00  INT 21 | 2000  2000  2000  2000  2000  0000  0000  4C00 | 0000  0000  0000  0000  0000  0000  2000  2000 | 0000  0008  000B  000C  000D  000E  000F  0012 | 0000  0010  0014  0012  0010  0012  0014  0014 |

在Debug中用T命令执行第二行语句mov ss, ax后，它的下一条指令mov sp, 10h并没有显示，这是因为Debug的T命令在执行修改寄存器SS的指令时，下一条指令也紧接着被执行。

下面以压栈图的形式展示栈操作的过程：

|  |  |
| --- | --- |
|  |  |
| add sp, 4后 | push ax后 |
|  |  |
| push bx后 | pop ax后 |
|  | |
| pop bx后 | |

上述过程中，push ax后，SP=SP-2，将ax的值送入SS:SP指向的内存单元；push bx后，SP=SP-2，将bx的值送入SS:SP指向的内存单元；pop ax后，先取出SS:SP指向内存单元处的一个字放入ax，然后SP=SP+2；pop bx后，先取出SS:SP指向内存单元处的一个字放入bx，然后SP=SP+2。

经过以上程序，交换了寄存器ax和bx中的值，IP每次增加的值是执行汇编指令对应的机器码的字节数。

实验报告四

一、实验目的：

学习计算机中的数制，记住常用字符的ASCII码。

二、实验任务：

1．下面的程序的功能是将mov ax, 4c00h之前的指令复制到内存0:200处，上机调试，跟踪运行结果。

|  |
| --- |
| assume cs: code  code segment  mov ax, cs  mov ds, ax ;将代码段作为数据段  mov ax, 0020h  mov es, ax ;(es)=0020h  mov bx, 0  mov cx, offset a  s: mov al, [bx]  mov es: [bx], al ;将ds: [bx]复制到es: [bx]  inc bx  loop s  a: mov ax, 4c00h  int 21h  code ends  end |

使用u命令进行反汇编，d命令查看0:200处内容并记录。

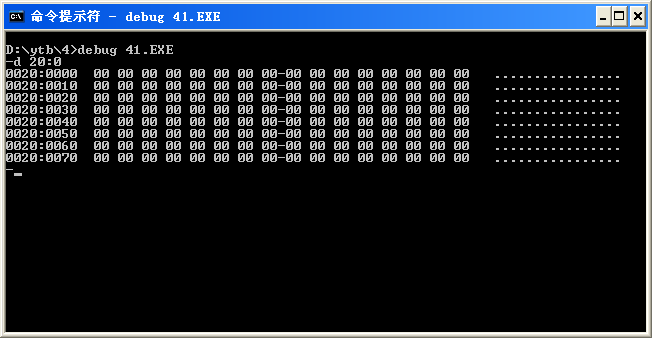
2．编程：向内存0:200-0:023f依次传送0-63（3fh）。程序只能使用9条指令，9条指令中包括mov ax, 4c00h，int 21h不包括伪指令。

3．编写一个汇编程序，要求从键盘输入一个小写字母，将其转换成大写字母在屏幕上显示出来。

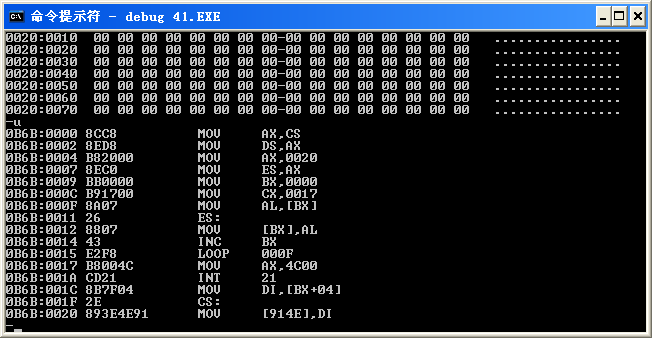
二、实验过程：

1．

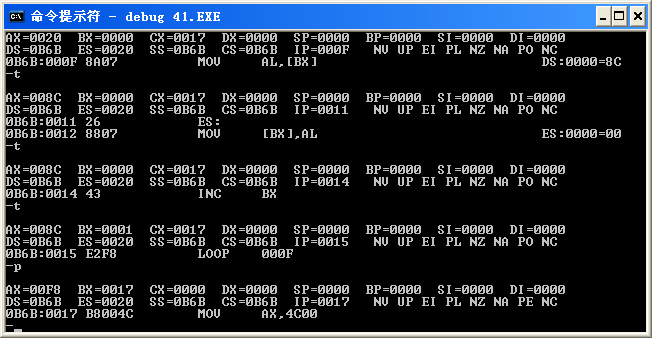
（1）开始时使用d命令查看内存0020:0处全为0：



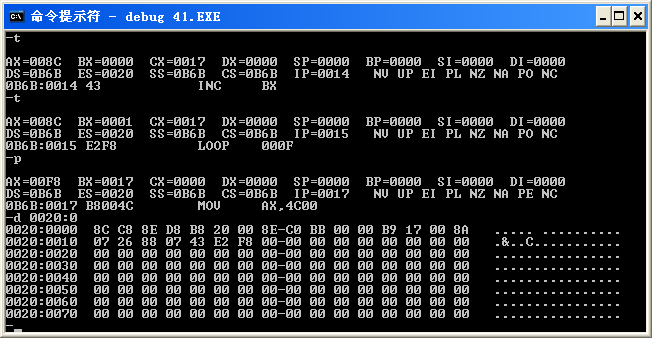
（2）使用u命令进行反汇编，记录机器码：



（3）连续使用t命令单步执行，执行到LOOP指令使用p命令跳过整个循环：



（4）再次查看内存，可以看到，和第（2）步中记录的机器码相同：



（4）该代码的流程图如下：

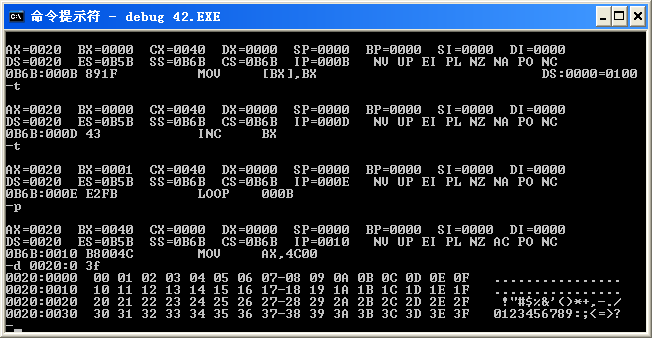


2．

（1）代码如下：

|  |
| --- |
| assume cs: code  code segment  mov ax, 0020h  mov ds, ax ;设置数据段地址为0020h  mov bx, 0  mov cx, 0040h ;循环64次  s: mov [bx], bx  inc bx  loop s  mov ax, 4c00h  int 21h  code ends  end |

（2）程序结束前使用d命令查看相应位置的内存：

（3）该代码的流程图如下：



3．

（1）进制和ASCII码的一些相关知识：

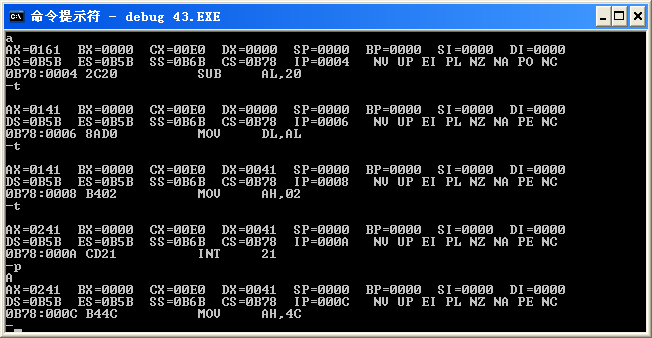




（2）代码如下：

|  |
| --- |
| STACK SEGMENT  DB 200 DUP(0)  STACK ENDS  CODE SEGMENT  ASSUME CS: CODE, SS: STACK  BEGIN:  MOV AH, 1  INT 21H ;从键盘输入一个小写字母放入AL  SUB AL, 20H ;小写字母转大写字母  MOV DL, AL  MOV AH, 2  INT 21H ;在屏幕上显示DL的内容  MOV AH, 4CH  INT 21H ;程序返回  CODE ENDS  END BEGIN |

（3）测试成功输入小写字母a输出大写字母A：



（4）该程序的流程图如下：



实验报告五

一、实验目的：

学习了解从键盘上输入字符、在显示器上输出字符的方法。

二、实验任务：

1．设计程序，要求从键盘上逐一输入字符，并在显示器上输出，当输入到“$”时，则停止操作。

2．编程：在已知BUF为首地址的字节存储区中，存放着一个以“$”作结束标志的字符串。编程在显示器上显示该字符，并要求将小写字母以大写字母的形式显示出来。

三、实验代码：

1．

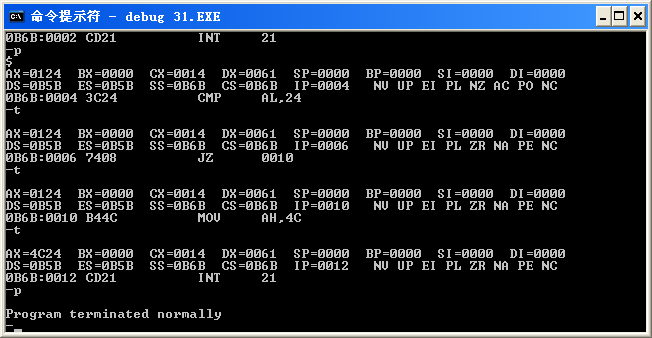
|  |
| --- |
| CODE SEGMENT  ASSUME CS: CODE  G1:  MOV AH, 1  INT 21H ;从键盘读入字符送入AL  CMP AL, '$'  JZ EXIT ;若字符为$结束输入  MOV DL, AL  MOV AH, 2 ;输出字符  INT 21H  JMP G1 ;跳转G1继续输入  EXIT:  MOV AH, 4CH  INT 21H  CODE ENDS  END G1 |

2．

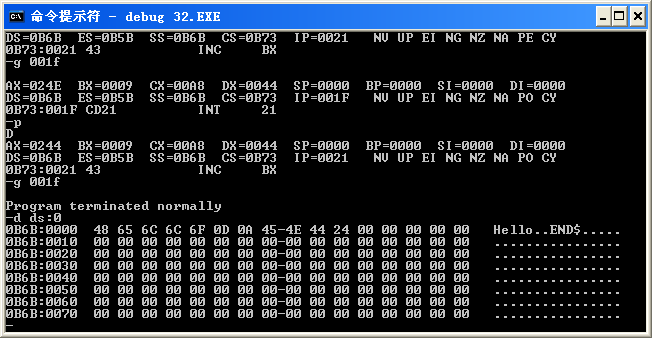
|  |
| --- |
| DATA SEGMENT  BUF DB 'Hello', 0Dh, 0Ah, 'END$'  DATA ENDS  STACK SEGMENT  DB 100 DUP (0)  STACK ENDS  CODE SEGMENT  ASSUME CS: CODE, DS: DATA, SS: STACK  BEGIN:  MOV AX, DATA  MOV DS, AX  LEA BX, BUF  LA:  MOV DL, [BX] ;(DL)=([BX])  CMP DL, '$'  JZ EXIT ;若DL中的字符是$则结束  CMP DL, 'a'  JB K  CMP DL, 'z'  JA K ;若DL中的字符不是小写字母，转K  SUB DL, 20H ;小写转大写  K:  MOV AH, 2  INT 21H ;显示字符  INC BX  JMP LA  EXIT:  MOV AH, 4CH  INT 21H  CODE ENDS  END BEGIN |

四、实验记录：

1．



2．显示缓冲区中的信息：

．

五、程序框图：

|  |  |
| --- | --- |
| 任务1流程图 | 任务2流程图 |