实验报告六

一、实验目的：

1．掌握汇编语言程序上机过程。

2．掌握汇编语言结构。

3．学习汇编语言的伪操作。

4．了解汇编程序、连接程序、DOS系统装入和返回功能。

5．掌握用DEBUG调试汇编语言程序的方法。

二、实验任务：

1．什么叫系统的功能调用？举例说明实现DOS系统功能调用的一般步骤。

2．输入两个数a、b，比较两个数的大小，若a>b，则显示’a>b’，若a=b则显示’a=b’，否则显示’a<b’。

3．在屏幕上显示一行提示信息，并将其存入内存缓冲区，屏幕显示问候信息，并输出姓名。

三、实验过程：

1．在多用户和多任务的环境下，与硬件有关的ROMBIOS资源只允许操作系统这个特殊用户使用，用户只可以使用INT 21H功能调用，包括字符输入、字符输出、磁盘控制、文件管理、记录操作、目录操作、动态存储分配等功能。

中断类型21H是DOS功能调用的主体，它提供了众多的非常强大的功能供用户调用，功能号由寄存器AH提供。在发INT 21H软中断之前，应该先准备好入口参数（也称调用参数），并将功能号送入AH寄存器中。

例如：由键盘输入单个字符

MOV AH, 1 ;系统调用功能号送入AH

INT 21H

输入字符的ASCII码在AL寄存器中，同时字符显示在屏幕上。

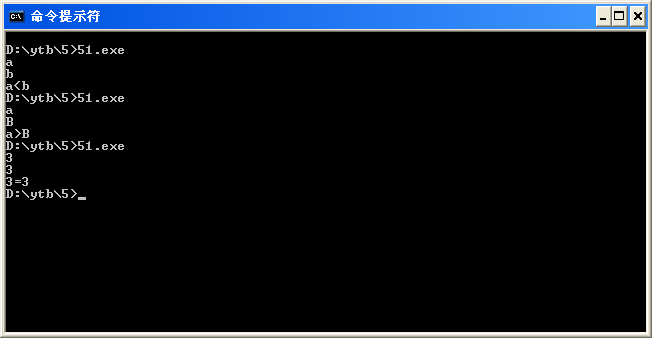
附表 键盘和显示器的DOS功能调用

|  |  |  |  |
| --- | --- | --- | --- |
| 调用号 | 功能 | 入口参数 | 出口参数 |
| 1 | 键入并显示一个字符 |  | 键入字符的ASCII码在AL中 |
| 2 | 显示器显示一个字符 | DL中置输出字符的ASCII码 |  |
| 5 | 打印机打印一个字符 | DL中置输出字符的ASCII码 |  |
| 8 | 键盘输入一个字符 |  | 键入字符的ASCII码在AL中 |
| 9 | 显示器显示字符串 | DS: DX置字符串首址，字符串以’$’结束 |  |
| 10(0AH) | 键入并显示字符串 | DS: DX置字符串首址，第1单元置允许键入的字符数(含一个回车符) | 键入的实际字符数在第2单元中.键入的字符从第3单元开始存放 |
| 11(0BH) | 检测有无键入 |  | 有键入AL=FFH，无键入AL=0 |

2．程序代码：

|  |
| --- |
| STACK SEGMENT STACK  DW 256 DUP(?)  STACK ENDS  CODE SEGMENT  ASSUME CS: CODE, SS: STACK  START:  MOV AH, 1  INT 21H ;使用调用号1输入字符送入AL中，并显示在屏幕上  MOV BH, AL ;用BH存放键盘输入的保存在AL中的字符  MOV AH, 2  MOV DL, 0DH  INT 21H  MOV DL, 0AH  INT 21H ;另起一行  MOV AH, 1  INT 21H ;从键盘输入另一个字符送入AL中  MOV BL, AL ;将字符保存在BL中  MOV AH, 2  MOV DL, 0DH  INT 21H  MOV DL, 0AH  INT 21H ;另起一行  MOV AH, 2  MOV DL, BH  INT 21H ;使用调用号2在屏幕上输出BH的值  CMP BH, BL ;比较BH和BL的大小  JG DISP2 ;若BH>BL，跳转到DISP2  JE DISP3 ;若BH=BL，跳转到DISP3  DISP1:  MOV DL, '<' ;将'<'保存在DL中  JMP EXIT  DISP2:  MOV DL, '>' ;将'>'保存在DL中  JMP EXIT  DISP3:  MOV DL, '=' ;将'='保存在DL中  EXIT:  INT 21H ;使用调用号2在屏幕上输出DL的值  MOV DL, BL  INT 21H ;使用调用号2在屏幕上输出BL的值  MOV AH, 4CH  INT 21H ;程序返回  CODE ENDS  END START |

该程序运行后会等待用户进行输入，输入一个字符a后自动换行等待用户输入第二个字符b，输入结束后将比较a、b的ASCII码值，然后输出大小比较的结果：



该程序的流程图如下：



3．使用DOS系统功能调用10号功能将从键盘接受字符串送到内存的输入缓冲区（由DS: DX指定缓冲区），要求预先定义一个缓冲区，缓冲区的第一个字节指定容纳的最大字符个数，由用户给出；第二个字节存放实际的字符总个数，由系统最后添入；从第三个字节开始存放从键盘接受的字符，直到ENTER键结束。9号功能可以在显示器上字符串，字符串从DS: DX开始，直到第一个’$’字符结束。

现在想要在屏幕上输出使用10号功能输入存储在buff处的字符串，有如下两种方法：

（1）取出buff第二个字节处存放的字符串的字符个数，定位到字符串的末尾并添加’$’，程序代码如下：

|  |
| --- |
| stack segment stack  db 64 dup(?)  stack ends  data segment  buff db 50, ?, 50 dup(?)  mes1 db 0dh, 0ah  db 'What is your name?:$'  mes2 db 0dh, 0ah  db 'hello!', '$'  data ends  code segment  assume cs: code, ds: data, ss: stack  start:  mov ax, data  mov ds, ax  ;在屏幕上输出mes1字符串  mov dx, offset mes1  mov ah, 9  int 21h  ;从键盘输入字符串存入buff中  mov dx, offset buff  mov ah, 0ah  int 21h  ;在屏幕上输出mes2字符串  mov dx, offset mes2  mov ah, 9  int 21h  ;定位到buff的第二个字节（存储的是输入的字符串的长度）  mov si, offset buff+1  mov bx, [si]      mov bh, 0 ;bx的高位设为0  mov byte ptr [bx+si+1], '$' ;在字符串结尾处添加'$'  ;在屏幕上输出buff处第三个字节开始存储的字符串  mov dx, offset buff+2  mov ah, 9  int 21h  mov ah, 4ch  int 21h  code ends  end start |

（2）将buff缓冲区从第三个字符开始全部填充为’$’，这样在输入后字符串的末尾便为’$’，修改代码如下：

|  |
| --- |
| stack segment stack  db 64 dup(?)  stack ends  data segment  buff db 50, ?, 50 dup('$') ;将缓冲区从第三个字符开始全部填充为'$'  mes1 db 0dh, 0ah  db 'What is your name?:$'  mes2 db 0dh, 0ah  db 'hello!', '$'  data ends  code segment  assume cs: code, ds: data, ss: stack  start:  mov ax, data  mov ds, ax  ;在屏幕上输出mes1字符串  mov dx, offset mes1  mov ah, 9  int 21h  ;从键盘输入字符串存入buff中  mov dx, offset buff  mov ah, 0ah  int 21h  ;在屏幕上输出mes2字符串  mov dx, offset mes2  mov ah, 9  int 21h  ;在屏幕上输出buff处第三个字节开始存储的字符串  mov dx, offset buff+2  mov ah, 9  int 21h  mov ah, 4ch  int 21h  code ends  end start |