2．解决除法溢出的问题

|  |
| --- |
| stack segment  dw 64 dup (?)  stack ends  code segment  assume cs: code, ss: stack  start:  ;设置栈段  mov ax, stack  mov ss, ax  mov sp, 64  ;调用测试  mov ax, 4240H  mov dx, 000FH  mov cx, 0AH  call divdw    mov ax, 4C00H  int 21H  divdw proc near ;32位除以16位无溢出  push bx ;保存bx的值  push ax  mov ax, dx  mov dx, 0  div cx  mov bx, ax ;保存商（最终结果的低16位）  pop ax  div cx ;得到结果的低16位  mov cx, dx ;转移余数到cx中  mov dx, bx ;最终结果的高16位    pop bx ;恢复bx的值  ret ;返回到调用处  divdw endp  code ends  end start |

3．数值显示

|  |
| --- |
| stack segment  dw 64 dup (?)  stack ends  data segment  db 10 dup (0)  data ends  code segment  assume cs: code, ss: stack  start:  mov ax, stack  mov ss, ax  mov sp, 64 ;栈段  mov ax, 12666  mov bx, data  mov ds, bx ;数据段  mov si, 0  call dtoc ;转换为以0结尾字符串存储在ds: si位置  mov dh, 8 ;行号  mov dl, 3 ;列号  mov cl, 2 ;绿色  call show\_str ;显示字符串  mov ax, 4C00H  int 21H  ;16位除以8位无溢出，ax被除数，cl除数，ax存储商，cl存储余数  divw proc near  push bx ;保存bx的值  mov bh, al ;保存al（低8位）的值  mov al, ah  mov ah, 0  div cl  mov bl, al ;保存商（最终结果的低8位）  mov al, bh ;恢复al（低8位）的值  div cl ;得到结果的低8位  mov cl, ah ;转移余数到cl中  mov ah, bl ;最终结果的高8位    pop bx ;恢复bx的值  ret ;返回到调用处  divw endp  dtoc proc near  push ax  push bx  push cx  push si  mov bl, 0 ;记录栈中元素个数  s1: mov cl, 0AH ;十进制数10  call divw ;除10取余  mov ch, 0 ;cx存储余数  push cx  inc bl ;栈中元素增加1  mov cx, ax  jcxz save ;商为0停止  jmp short s1  save:  mov cl, bl  mov ch, 0 ;循环栈中元素个数的次数  s2: pop ax  add ax, 30H ;转换为ASCII码  mov word ptr [si], ax  inc si ;定位到下一个字  loop s2  mov word ptr [si], 0  pop si  pop cx  pop bx  pop ax  ret  dtoc endp  ;注：show\_str子程序参考了一篇CSDN博客中的代码  show\_str proc near  push bx  push cx  push es  push si  push di  mov ax, 0b800h  mov es, ax  mov al, 0a0h ;一行的总列数160字节  dec dh ;行号减1，因为是从0开始的  mul dh ;计算行开始偏移地址  mov bx, ax  mov al, 2  mul dl ;计算列  sub ax, 2 ;列也是从0开始，而且一个字符占两个字节  add bx, ax ;求出开始位置  mov di, 0  mov al, cl  mov ch, 0 ;高8位为0  s: mov cl, [si] ;判断是否到了字符结束  jcxz ok  mov es: [bx+di], cl  mov es: [bx+di+1], al  inc si  add di, 2  jmp short s  ok: pop di  pop si  pop es  pop cx  pop bx  ret  show\_str endp  code ends  end start |