Turbo C 语言与汇编语言混合编程探讨

云南省军区自动化站 李晓华

摘要：本文以 *Turbo C* 语言和宏汇编语言为例，说明 *Turbo C* 与汇编语言的混合编程技术，阐述了两种语言间的调用协定。最后给出了一个程序的框架。

一、概述

为了提高程序执行速度和效率，实现某些用 Turbo C 无法做到的机器语言操作，同时为了使用汇编语言写好的子程序，所以必须考虑 Turbo C 与汇编语言混合编程问题。虽然 Turbo C 语言能够产生快速、紧凑的目标代码，但没有一种编译所产生的目标代码能象汇编语言所编写的程序那样高速和紧凑。

Turbo C 与汇编语言的混合编程有 2 种形式：一种是采用传统的汇编语言接口；另一种是采用行间汇编语言。本文介绍的是前一种方法。后一种方法有关书籍已有过介绍。

二、汇编语言接口

编写低级系统例程的传统方法是用汇编语言写出，然后将模块连结到 Turbo C 程序上。由于汇编语言的例程与内存模型有关，所以首先必须掌握有关内存模型的关系。

1.内存模型与说明符

Turbo C 及多数 8086 系列用的编译器根据寻址方式的不同，将数据指针及函数指针划分 near(近)、far(远)、huge(巨)。Turbo C 为程序员提供了六种不同的内存模型。具体方式请参考有关书籍。

2.调用协定

调用协定是 Turbo C 编译程序用来传递信息给函数(汇编语言编写的)，以及从函数返回传递的方式。通常解决方法是使用 CPU 的内部寄存器或系统堆栈来传递函数间的信息。标准的汇编接口方式包括下列步骤：

(1)建立过程。正文段、段、组描述，用 PUBLIC 说明标号是公有的，用 extrn 说明该程序的局部数据或过程(通常放在段外)。

(2)进入过程。BP 称为帧指针，用于存取位于栈中的参数和局部数据，SP 的值随着参数的压入栈而改变。首先保存 BP，然后把 SP 的值压入 BP，以获得进入过程时栈指针的值。

```
PUSH BP
MOV BP, SP
```

(3)分配局部数据。汇编过程可使用与 Turbo C 语言实现局部数据相同的技术。只须简单地在过程中的第 3 条指令中减少 SP 的值，就可以建立局部数据空间。

```
push bp
mov bp, sp
sub sp,space        ; space 是局部数据的总字节数
例:
push    bp
mov     bp sp
mov     word ptr [bp-2],0
mov     mord ptr [bp-4],0
```

上例中使用了 2 个局部变量，每个变量大小都是 2 个字节。因为局部数据总长度为 4 字节，所以 SP 减 4。

(4)保存寄存器。任何高级语言的调用过程都要保存 si,di,ss,ds 的值。

```
push    bp
push    bp,sp
sub     sp,space
push    si
push    di
.....
```

(5)存取参数。在建立了过程的框架指针、分配了局部数据空间和需要保存的寄存器值压栈后，就可编写程序的主体了。在存取参数时，它们的压栈顺序见表 1。由于压栈的大小与数据类型有关，所以数据大小与数据类型有关，见表 2。

表 1 参数的压栈顺序结构

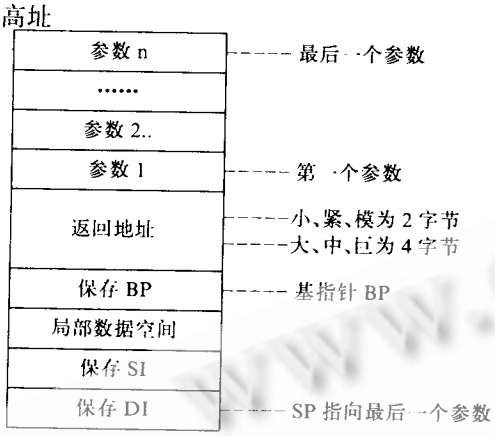

表 2　数据类型与字节数大小

参数类型	字节数
char	2
int	2
long	4
double	8

(6)返回值。数据值的返回是放在下列寄存器中：

数据长度	返回寄存器
1 字节	al
2 字节	ax
4 字节	dx(高位字):ax(低位字)

(7)退出过程。若 ss、ds、si、di 已保存，则按它们的相反过程退出。若在开始时分配了空间，则用 MOV SP,BP 恢复。最后用 POP BP,RET 恢复 BP 返回调用程序。

三、Turbo C 语言调用汇编程序

下面给出一个通用的 Turbo C 语言调用汇编程序的框架程序：

它所采用参数的传递方式不用堆栈传递，而是利用数据结构(struc)进行参数传递。数据结构定义如下：

```
sfram struc                ; stack frame template
baseptr     dw   ?         ; base pointer
retad       dw   ?         ; molde (samll) (return address)
                           ; Medium, Large, Huge: replace
                           'dw' with 'dd'
str_ad      dw   ?         ; string address
str_attrib  dw   ?         ;string attrib
row         dw   ?         ; starting row
col         dw   ?         ; starting col
sframe ends
/*                                                    */
/*   filename    writvide.prj                         */
/*   生产writvide.exe执行文件                         */
/*   writvide.exe:    writvide.c                      */
/*   writvide.obj (.asm)                              */
/*                                                    */
writvide
writvide.obj
```

```
/*                                                    */
/*   filename writvide.c                              */
/*   通过调用由汇编语言编写的函数(writvide.obj)       */
     显示字符串。                                     */
/*   调用格式: writvide (参数1, 参数2, 参数3, 参数4)  */
/*   其中:参数1=字符串                                */
/*        参数2=颜色                                  */
/*        参数3=X坐标                                 */
/*        参数4=Y坐标                                 */
/*                                                    */
#include <stdio.h>
extern void writvide();
main()
{
char *vp="这是一个 Turbo C 语言与汇编语言混合编程的例子程序 this is a ...";
writvide (vp, 0x0f,5,7);
{
```

```
;
;   mname    writevide.asm
;   供 C 语言调用 writvide.obj: masm writvide / MX;
;   this procedure writes a null terminated string
;    DIRECTLY TO VIDEO memroy
;   C call void writvide (char * str, int str__attrib,
;    int row, int col);
;
__text segment byte public 'code'
dgroup group __data
assume cs: __text, ds: dgroup, ss:dgroup
__text   ends
__data   segment word public 'data'
hzm      db 0
__data   ends
__text   segment byte public 'code'
         public__ writvide
__writvide proc near
sframe struc                ; stack frame template
baseptr     dw     ?     ; base pointer
retad       dw     ?     ; molde (samll)
str__ad      dw     ?     ; string address
str__attrib  dw     ?     ; string attrib
row          dw     ?     ; starting row
col          dw     ?     ; starting col
sframe   ends
frame    equ      [bp-baseptr]
         push     bp
         mov      bp,sp
         sub      sp,baseptr
         push     ds
         push     es
         push     di
         push     si
;
         mov      ax, data
         mov      es,ax
         push     es
         pop      ds
         mov      ah,1ah
         mov      al,1
         int      10 h
;
         mov      ah,5
         mov      al,0
         int      10h
         call     clear
;
         call     posn
         mov      si,frame.str_ad      ; ds: si point to start of
                                       string
         mov      ah,byte ptr frame.str_attrib; ah: string attrib
again:
         lodsb
         cmp      byte ptr [si],0      ; test for null termination
         je       eofl
         call     dispchar             ; display a BYTE char
         jmp      again
;                                                    ; exit
eofl:
         mov      ah,5
         mov      al,0
         int      10h

         pop      si
         pop      di
         pop      es
         pop      ds

         mov      sp,bp
         pop      bp
         ret

posn     proc     near
         mov      ax, 0b800h
         mov      es,ax
         mov      di,frame.row         ; starting offset in video
         mov      ax,di    ;memory = (row * 160)+(col * 2);
         mov      cl,7
         sh1      di,cl
         mov      cl,5
         sh1      ax,cl
```

```
        add     di,ax
        mov     ax,frame.col
        shl     ax,1
        add     di,ax
        ret
posn    endp
;
clear   proc    near
        mov     ah,7
        mov     al,0
        mov     cx,0
        mov     dx,184fh
        mov     bh,7
        int     10h
        ret
clear   endp
;
dispchar proc   near ;
        cmp     al,0a1h
        jze     hzcl
        stosw
        dec     di
        push    es
        push    ax
        mov     ax,0b000h
        mov     es,ax
        mov     byte ptr es:[di-1],0
        pop     ax
        pop     es
        inc     di
        mov     hzm,0
        ret
hzcl:
        cmp     hzm,0
        jnz     whz
        mov     hzm,al
        ret
whz:
        mov     zh,hzm
        and     ax,7f7fh
        sub     ax,2121h
        cmp     ah,15
        jb      w1
        sub     ah,6
w1:
        mov     cs,ax
        mov     ax,94
        mul     ch
        xor     ch,ch
        add     ax,cx
        add     ax,256
        or      ah,40h
        stosw
        dec     di
        push    es
        push    ax
        mov     ax,0b000h
        mov     es,ax
        pop     ax
        mov     byte ptr es:[di-1],ah
        pop     es
        inc     di
        inc     di
        inc     di
        mov     hzm,0
        ret
dispchar endp
;
 writvide endp
 text ends
        end
```

说明：本程序在 CEGA 卡上通过。