Desarrollo de Programas Residentes en base a un TSR genérico

Ing. Eduardo Vega Alvarado
Jefe del Departamento de Laboratorios Ligeros del CINTEC-IPN

El presente artículo inicia una serie de trabajos sobre tópicos relacionados con programación, incluyendo tanto al desarrollo de algoritmos o rutinas específicas como la interacción de estos programas con el sistema operativo.

Introducción

Cuando se desea ejecutar un programa en el entorno del sistema operativo **MS-DOS** (**M**icro**S**oft **D**isk **O**perating **S**ystem), generalmente el cargador de programas del sistema busca espacio libre en la memoria para colocar el código solicitado, transfiere control al mismo y al terminar el procesamiento libera el área de memoria asignada a dicho código **[3]**; sin embargo, existe cierta clase de programas cuya ejecución no incluye esta última etapa. En esencia, todo programa que permanece cargado en memoria después de que termina su ejecución (incluso aún cuando se inicie la ejecución de otro programa), se denomina **TSR** (**T**erminate and **S**tay **R**esident, termina y permanece residente) **[4]**.

TSR Genérico

El principal problema a enfrentar cuando se escribe un programa TSR es el proporcionar al usuario la forma de activarlo en el momento que lo desee; esto es obvio, dado que aunque al código originalmente en memoria se le adicione una sección extra, esta porción no se ejecutará si no existe forma de invocarla. No basta únicamente con asignar un área permanente en memoria para la colocación del TSR, sino que es necesario además asegurar algún medio para transferirle control. En forma genérica **[2] [4]**, un TSR consta de tres partes principales, a saber:

1. El programa o rutinas que se desean ejecutar.

2. El conjunto de instrucciones que coloca al TSR en memoria e instruye al sistema operativo para que lo considere residente (Inicialización).

3. El código que implementa la vía de acceso o de ejecución al TSR.

La distribución usual de estas partes en un programa real se muestra en la **figura 1.**

A continuación se desarrolla un programa ejemplo, con el cual se explicará el proceso de generación de un TSR, indicando además qué secciones del mismo corresponden al modelo genérico y cuáles son propias de este caso específico. La ejecución del programa genera un

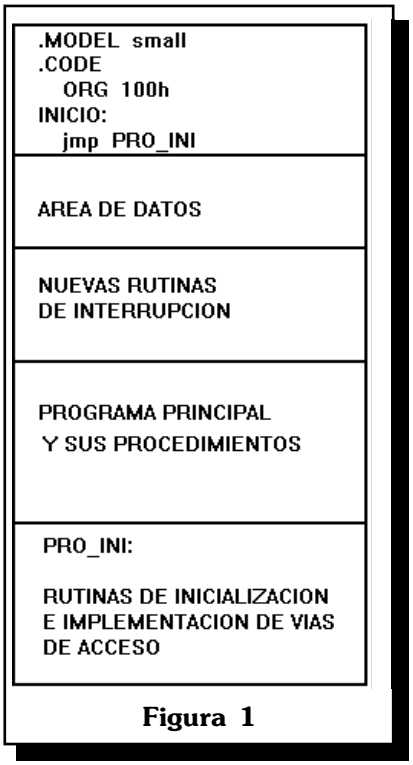

Figura 1

reloj, el cual aparece en la parte superior derecha de la pantalla desplegándose en forma digital; el listado se muestra en la anexo 1. Este programa ejemplo fue probado compilándolo con **Macro Assembler** ᴿ **[1]**, **MASM**, versión 5.1 de **Microsoft, Inc**., aunque puede también manejarse sin cambios con **Turbo Assembler** ᴿ, **TASM**, de **Borland International, Inc**.

Zona de Datos del Programa

Normalmente, los TSR comienzan con una instrucción de salto

```
1   ; TSR que Genera un Reloj en Pantalla
2   ; Programó: Ing. Eduardo Vega Alvarado
3   ;-------------------------------------------------------
4   .model   small
5   .code
6       org    100h
7
8   Inicio:
9       jmp    Instalar
10
11  ; Area para Datos del TSR
12  ;-------------------------
13  ; Vectores de interrupción originales
14      i1C             label   dword       ;int 1Ch
15      off_i1C     dw      ?
16      seg_i1C         dw      ?
17      i10             label   dword       ;int 10h
18      off_i10         dw      ?
19      seg_i10         dw      ?
20      i28             label   dword       ;int 28h
21      off_i28         dw      ?
22      seg_i28         dw ?
23
24      vid_act     db      0       ;bandera de int 10h activa
25      activo          db      0       ;tsr en ejecución
26      off_indos       dw      ?       ;segmento y offset de la bandera
27      seg_indos       dw      ?       ;de dos activo (indos)
28      err_act         dw      ?       ;offset de la bandera de error crítico
29      cursor1         dw      ?       ;posición y tamaño
30      cursor2         dw      ?       ;del cursor
31      sp_sist     dw      ?       ;sp y ss originales
32      ss_sist         dw      ?
33                      db  255 dup(?);pila propia
34      tsr_pil         db      ?       ;inicio de pila propia
35      horas10         db      ?       ;cadena para desplegar la hora
36      horas           db      ?
37                      db      ":"
38      mins10          db      ?
39      mins            db      ?
40                      db      ":"
41      segs10          db      ?
42      segs            db      ?
43
44  ; Rutina modificada para la int 1Ch
45  ;-----------------------------------
46  Sust1C      proc    near
47      mov         activo,18
48      pushf                       ;simula llamado a INT
49      call        i1C             ;invoca int 1Ch original
50      push        di
51      push        es
52
53  ; Checa si está activa int 10h, la bandera indos o se
54  ; atiende un error crítico. En caso afirmativo finaliza
55  ; rutina int 1Ch, en caso contrario ejecuta Reloj
56      cmp         vid_act,0       ;¿int 10h activa?
57      jne         Fin1C
```

hacia la zona de inicialización del programa, reservando en la parte inicial del segmento de código una zona para manejo de las estructuras de datos requeridas por el programa. En el programa ejemplo esta zona de almacenamiento abarca las líneas 11 a 42.

Inicialización del TSR y Acceso al Programa Residente

La sección correspondiente a la colocación en memoria del programa y a los cambios necesarios para instrumentar una vía de acceso al mismo se sitúa como la parte final del código, dado que su ejecución se realiza solo una vez; de esta forma, en el momento en que la rutina queda instalada esta área se descarta. En el listado anexo, la inicialización y colocación ocupa las líneas de código 222 a 275; como puede observarse, primeramente se interceptan o cambian los vectores de las interrupciones que sirvan como vía de acceso a la rutina residente **[2]**, o cuyo funcionamiento pudiera ser afectado por la misma (servicio 25h, int 21h). También se conservan las direcciones originales de las interrupciones empleadas (servicio 35h, int 21h), con el fin de permitir la invocación de los servicios correspondientes a esas rutinas, así como la desactivación del TSR en un momento dado.

Para el caso del ejemplo se manejan como vías de acceso las interrupciones 1Ch y 28h **[1] [2]**; ambas rutinas se producen automáticamente 18.2 veces por segundo, siendo generada la primera por la máquina y la segunda por el planificador **(scheduler)** del sistema operativo. Así mismo, se modifica el vector de la int 10h (manejo de video); esto debido a que la ejecución del programa en forma

```
58      mov     di,off_indos        ;¿bandera indos activa?
59      mov     es,seg_indos
60      cmp     byte ptr es:[di],0
61      jne     Fin1C
62      mov     di,err_act          ;¿atención a error crítico?
63      cmp     byte ptr es:[di],0
64      jne     Fin1C
65      pop     es
66      pop     di
67      call    Reloj               ;ejecuta la rutina principal del TSR
68      iret
69  Fin1C:
70      pop     es
71      pop     di
72      dec     activo
73      iret                        ;return to MS-DOS
74  Sust1C      endp
75
76  ; Rutina modificada para la int 10h
77  ;-------------------------------------
78  Sust10      proc    near
79      pushf                       ;simula llamado a INT
80      inc     vid_act
81      call    i10                 ;invoca int 10h original
82      dec     vid_act
83      iret
84  Sust10      endp
85
86  ; Rutina modificada para la int 28h
87  ;-------------------------------------
88  Sust28      proc    near
89      pushf                       ;simula llamado a INT
90      call    i28                 ;invoca int 28h original
91      cmp     activo,0
92      je      Fin28
93
94  ; Checa si está activa int 10h o se atiende un error
95  ; crítico. En caso afirmativo finaliza rutina int 28h
96  ; en caso contrario ejecuta Reloj
97      cmp     vid_act,0           ;int 10h activa?
98      jne     Fin28
99      push    di
100     push    es
101     mov     es,seg_indos        ;atención a error crítico?
102     mov     di,err_act
103     cmp     byte ptr es:[di],0
104     pop     es
105     pop     di
106     jne     Fin28
107     mov     activo,0
108     call    Reloj               ;ejecuta la rutina principal
109 Fin28:
110     iret
111 Sust28      endp
112
113 ; Rutina principal del TSR. Realiza el llamado a la lectura
114 ; de la hora y efectúa el despliegue de la misma
```

concurrente con la activación de la int 10h puede generar basura en el despliegue. Cabe mencionar que existe una interrupción, la 8h, que también se genera 18.2 veces por segundo; la diferencia entre esta interrupción y la 1Ch radica en las rutinas de servicio asociadas a cada una de ellas. La interrupción 8h maneja un determinado código para actualizar la hora del sistema (almacenada en el área de bios), mientras que la única instrucción de la interrupción 1Ch es el retorno de la misma, **iret**. Por esta razón se recomienda interceptar para aplicaciones de usuario la int 1Ch, evitando así el provocar disturbios en el manejo de la hora interna del sistema.

Posteriormente se leen los apuntadores a **indos** e **indos2** (servicios 34 y 5d-06, int 21h) **[3]**; estas banderas son manejadas por el sistema operativo para indicar si se está procesando una llamada a MS-DOS o si se ejecuta una rutina de atención por error crítico, respectivamente. Las banderas indos e indos2, al igual que los servicios de la int 21h necesarios para obtenerlas forman parte del llamado MS-DOS no documentado **[3]**; se ha denominado así a un conjunto de interrupciones, servicios y estructuras de datos que, pese a estar implementadas en el sistema, no han sido aceptadas como existentes por parte de Microsoft. Si bien su empleo es a riesgo del programador, su aplicación es de dominio casi público entre los diseñadores de software, especialmente en los casos de programas TSR o cuando se requiere un manejo especializado de dispositivos periféricos, siendo utilizadas inclusive en los programas mismos de Microsoft, tales como Windows ® y Excel ®.

Por último, con auxilio de la etiqueta **final** se determina el nú-

```
115 ;-------------------------------------------------------------------
116 Reloj       proc     near
117 ;establece el stack local y salva registros del DOS
118     cli
119     mov      sp_sist,sp          ;guarda apuntadores a
120     mov      ss_sist,ss          ;la pila del sistema y
121     push     cs                  ;establece pila propia
122     pop      ss
123     mov      sp,offset tsr_pil
124     sti
125     push     ax                  ;salva registros
126     push     bx                  ;del sistema
127     push     cx
128     push     dx
129     push     si
130     push     di
131     push     ds
132     push     es
133     push     bp
134     mov      ah,0fh              ;¿trabajando en modo gráfico?
135     int      10h
136     cmp      al,3
137     jbe      Pos_cursor
138     cmp      al,7
139     je       Pos_cursor
140 Termino:
141     pop      bp                  ;modo gráfico activo, por
142     pop      es                  ;lo que termina rutina
143     pop      ds
144     pop      di
145     pop      si
146     pop      dx
147     pop      cx
148     pop      bx
149     pop      ax
150     cli
151     mov      ss,ss_sist
152     mov      sp,sp_sist
153     sti
154     ret
155 Pos_cursor:                      ;modo texto, continúa despliegue
156     mov      ah,03               ;lee posición original del cursor
157     int      10h                 ;y la almacena en área de datos
158     mov      cursor1,dx
159     mov      cursor2,cx
160     call     Tiempo              ;lee hora del sistema
161     push     cs
162     pop      ds
163     mov      ah,016              ;cambia atributo de cursor
164     mov      cx,1000h
165     int      10h
166     mov      ah,02               ;coloca cursor en la esquina
167     mov      dh,0                ;superior derecha de la pantalla,
168     mov      dl,70               ;en (0,70)
169     int      10h
170     mov      si,offset horas10   ;apuntador a cadena
171     mov      cx,8                ;para desplegar
```

mero de párrafos que conforman al TSR, dejándose a éste permanente en memoria (servicio 31, int 21h). Como ya se había mencionado antes, en el código residente no se incluye el área de inicialización.

Declaración y Manejo de las Nuevas Rutinas de Interrupción

Uno de los principales detalles que debe respetar la nueva rutina de interrupción es lo que se conoce como encadenamiento **(chaining) [2] [4]**. El encadenamiento consiste en que el procedimiento nuevo invoque a la interrupción original, ya que puede darse el caso de que se hubieran montado anteriormente otros programas residentes que utilicen dicha interrupción. Dado que el llamado de la rutina antigua es por medio de **call [1]**, es necesario eje-cutar primero la instrucción **pushf** para simular una invocación por interrupción y compensar así el regreso de la rutina, el cual está marcado por **iret**. En el listado anexo, las nuevas rutinas de interrupción ocupan las líneas 44 a 111.

Rutina Principal

Se denomina rutina principal al código que genera la acción que se desea realice el TSR (por ejemplo, desplegar la hora en pantalla, como en el listado anexo). Independientemente de su función, esta rutina debe cumplir algunas reglas no escritas, entre las que destaca el conservar los registros en su estado inicial, almacenándolos en la pila **(stack)**. Lo anterior obedece a que en esos registros se conserva parte del estado del programa que se suspendió para dar paso al TSR; en la medida de lo posible es conveniente implementar una zona de

```
Desplegar:
    lodsb                       ;muestra cadena en pantalla,
    mov     ah,0ah              ;escribiendo un carácter a la vez
    push    cx
    mov     cx,1
    int     10h
    pop     cx
    inc     dl
    mov     ah,02               ;avanza cursor una posición
    int     10h
    loop    Desplegar
    mov     ah,18               ;regresa cursor a posición original
    mov     cx,cursor2          ;y restablece registros del sistema
    int     10h
    mov     ah,2
    mov     dx,cursor1
    int     10h
    jmp     Termino
Reloj   endp

; Rutina para leer la hora del sistema, convertirla
; a código ASCII y almacenarla en el área de datos
; del TSR
;-------------------------------------------------
Tiempo  proc    near
    mov     ah,2ch
    int     21h                 ;lectura de la hora
    mov     bl,10
    xor     ah,ah
    mov     al,ch               ;horas en ch
    div     bl
    or      ax,3030h
    mov     horas10,al
    mov     horas,ah
    xor     ah,ah
    mov     al,cl               ;minutos en cl
    div     bl
    or      ax,3030h
    mov     mins10,al
    mov     mins,ah
    xor     ah,ah
    mov     al,dh               ;segundos en dh
    div     bl
    or      ax,3030h
    mov     segs10,al
    mov     segs,ah
    ret
Tiempo  endp
final   db  "$"     ;marca para fin de TSR

; Rutina de instalación en memoria del TSR. También
; modifica los vectores de interrupción utilizados
; o afectados por el TSR, para apuntar a los nuevos
; manejadores.
;-------------------------------------------------
Implementa  proc    near
assume  ds:@code        ;variables en este segmento
```

pila local en el área de datos del TSR, para evitar conflictos tales como desbordamiento si se emplea el stack convencional del sistema. En consecuencia, previo a la terminación de la rutina principal se deben restaurar los registros, siguiendo la operación inversa.

La rutina principal, al igual que las nuevas rutinas de interrupción, deben respetar la no reentrancia **(reentry) [2]** del sistema operativo. La no reentrancia consiste en que no se puede efectuar una llamada al sistema en tanto no finalice la llamada anterior, ya que de lo contrario se pueden generar fallas ocasionadas por el frágil manejo de MS-DOS para preservar su estado actual al momento de una interrupción. Este problema se maneja en el programa ejemplo mediante el uso de la bandera indos para detectar si existe una llamada al sistema activa y no ejecutar el TSR en tal caso. Por esta misma razón para el despliegue de la cadena generada con la hora se utiliza la int 10h servicio 0ah, que muestra un carácter a la vez, para no emplear la int 21h servicio 9h, que permite desplegar la cadena completa.

Como consecuencia de la no reentrancia del sistema operativo, se debe evitar activar el TSR cuando se presenta una condición de error crítico; estos errores se producen cuando el DOS necesita utilizar un dispositivo periférico pero este no se encuentra listo o disponible, activándose entonces la int 24h **[3]**, conocida como manejador de errores críticos. En el listado anexo se comprueba esta condición por medio de la bandera indos2, permitiéndose o no la ejecución del TSR, dependiendo del valor de esa bandera.

La rutina principal del programa ejemplo se explica línea por

```
229
230 Instalar:
231     mov     ah,34h              ;localiza bandera indos
232     int     21h
233     mov     off_indos,bx        ;offset y segmento
234     mov     seg_indos,es        ;de indos
235     mov     ah,5dh              ;localiza bandera de atención
236     mov     al,6h               ;a error crítico y copia apuntador
237     int     21h                 ;en el área de datos
238     push    cs
239     pop     ds
240     mov     err_act,si
241
242  ; Sustitución de vectores de interrupción afectados,
243  ; conservando los apuntadores originales para efectos
244  ; de restauración por terminación del TSR
245     mov     ax,351Ch            ; int 1Ch
246     int     21h
247     mov     off_i1C,bx
248     mov     seg_i1C,es
249     mov     ax,251Ch
250     mov     dx,offset Sust1C
251     int     21h
252     mov     ax,3510h            ; int 10h
253     int     21h
254     mov     off_i10,bx
255     mov     seg_i10,es
256     mov     ax,2510h
257     mov     dx, offset Sust10
258     int     21h
259     mov     ax,3528h            ; int 28h
260     int     21h
261     mov     off_i28,bx
262     mov     seg_i28,es
263     mov     ax,2528h
264     mov     dx,offset Sust28
265     int     21h
266
267  ; Cálculo de longitud del TSR y asignación de
268  ; espacio permanente en memoria
269     mov     dx,(offset final-@code+15)    ; No. de bytes
270     mov     cl,4h
271     shr     dx,cl                         ; No. de párrafos
272     mov     ax,3100h
273     int     21h
274  Implementa  endp
275     end Inicio
```

línea en el listado anexo (líneas 113 a 218); no se hace mayor hincapié en la misma puesto que puede ser sustituida en el TSR genérico descrito por cualquier otro procedimiento que se desee.

Consideraciones Finales

Dentro de las consideraciones finales se encuentran las correspondientes a la desactivación del TSR. La desactivación incluye dos tareas principales, la primera es interceptar los vectores de las interrupciones ocupadas de tal forma que tomen los apuntadores originales (los cuales previamente se guardaron en el área de datos del TSR) ocupando para ello la int 21h servicio 25h. La segunda tarea presenta mayor complejidad, y consiste en liberar el área de memoria ocupada por el TSR, después de su desactivación; debido a esta complejidad implícita, se considera conveniente manejar este tópico en un artículo posterior.

Finalmente, debe recordarse que si bien existen características del sistema no documentadas cuya utilidad es manifiesta, su uso se debe restringir a aquellas situaciones en las que constituyan la única opción dado que, por ser información no oficial, no se puede asegurar su no modificación o incluso su conservación en versiones posteriores del sistema operativo MS-DOS, lo que puede ocasionar problemas futuros de compatibilidad.

Bibliografía

[1] Barkakati, Nabajyoti & Hyde, Randall. *"Microsoft Macro Assembler Bible"*. SAMS. 1992.

[2] Holzner, Steven. *"Advanced Assembly Language"*. Peter Norton Computing, Inc. 1991.

[3] Schulman et Al. *"Undocumented DOS"*. Addison Wesley Publishing Company, Inc. 1990.

[4] Simrin, Steven. *"MS-DOS Bible"*. Howard W. Sams & Company. 1989.