Microprocesadores de la línea Intel - Anexo

por Darío Alejandro Alpern

Otros programas de ejemplo en modo protegido

```
; Realizar un programa en formato .COM que pase a modo protegido.
; Debe haber un segmento de datos que ocupe el primer MB de memoria.
; Llenar las posiciones de memoria 0B8000h-0B87FFh con la secuencia
; 31h, 07h, 31h, 07h, ... (esta es la zona del buffer de video).
; Al finalizar, volver a modo real y terminar el programa.
;
; Hecho por Dario Alejandro Alpern el 13 de abril de 1998.
;
CANT_DESCRIPTORES_GDT EQU 2 ; Cantidad de descriptores en la GDT.

          .386P           ;Este programa corre en 80386 o uP posterior.
                          ;Se agrega la "P" ya que se van a ejecutar
                          ;instrucciones privilegiadas.
codigo    segment use16   ;El segmento es de 16 bits.
                          ;Todos los programas DOS en 386 deben tener USE16.
          assume cs:codigo,ds:codigo
          org 100h        ;Todo programa .COM debe tener esta directiva ya
                          ;que los primeros 256 bytes del segmento estan
                          ;reservados para el Program Segment Prefix (PSP).
comienzo: jmp short inicio ;Saltear la zona de datos.
imagen_GDTR label fword   ;Indica que la imagen del GDTR ocupa 6 bytes.
                          ;La directiva LABEL no reserva espacio en memoria.
limite_GDTR dw CANT_DESCRIPTORES_GDT*8-1
                          ;El limite del GDTR es uno menos que la longitud
                          ;de la GDT (Tabla de Descriptores Globales).
base_GDTR dd 0            ;Se llena en tiempo de ejecucion.

GDT       db 8 dup (0)    ;El primer descriptor de la GDT nunca se usa.
          dw 0FFFFh       ;Bits 15-0 del limite del segmento de datos.
          dw 0000h        ;Bits 15-0 de la base del segmento de datos.
          db 00h          ;Bits 23-16 de la base del segmento de datos.
          db 10010010b    ;Byte de derechos de acceso.
                          ;   Bit 7 = Present (1 = si)
                          ;   Bits 6-5 = Descriptor Privilege Level (00).
                          ;   Bit 4 = Segment Descriptor (1 = datos o codigo).
                          ;   Bit 3 = Executable (0 = datos).
                          ;   Bit 2 = Expansion Direction (0: offset<=limite).
                          ;   Bit 1 = Writeable (1 = se puede escribir).
                          ;   Bit 0 = Accessed (0 = el segmento no se accedio).
          db 0Fh          ;Byte miscelaneo.
                          ;   Bit 7 = Granularidad (0 = limite x1).
                          ;   Bit 6 = Default Instruction (0 = 16 bits).
                          ;   Bits 5-4 = No usados.
                          ;   Bits 3-0 = Bits 19-16 del limite.
          db 00h          ;Bits 31-24 del limite.

inicio:
;
; Lo primero que hay que hacer es inicializar el valor base de GDTR.
; El valor absoluto depende de donde el DOS puso el programa en memoria,
; por eso hay que leer el registro de segmento CS para saber cual es el
; valor del segmento donde se encuentra la Global Descriptor Table.
;
          mov ax,cs               ; AX = Segmento de la GDT.
          mov bx,offset GDT       ; BX = Offset de la GDT.
          movzx eax,ax            ; Extender a 32 bits.
          movzx ebx,bx
          shl eax,4               ; Base GDT = Segmento GDT * 16 + Offset GDT
          add eax,ebx             ; EAX = Base de la GDT.
          mov base_GDTR,eax       ; Almacenar la base de la GDT.
          cli                     ; Deshabilitar interrupciones.
          lgdt imagen_GDTR        ; Cargar el GDTR con la base y limite de GDT.
          smsw ax                 ; Leer la Machine Status Word (MSW).
          or al,1                 ; Poner a uno el bit Protection Enable (PE).
          lmsw ax                 ; Almacenar la MSW. En este momento el uP
```

```
66                                      ; corre en modo protegido.
67          jmp short $+2               ; Vaciar la cola de instrucciones.
68          mov ax,8                    ; AX = Selector del segmento de datos.
69          mov ds,ax                   ; Cargar el nuevo segmento de datos.
70          mov edx,0B8000h             ; Direccion inicial donde se van a poner
71                                      ; los datos.
72          mov cx,800h/2               ; Cantidad de palabras (words) a poner.
73 ciclo_llenado:
74          mov word ptr [edx],0731h ; Dato a poner en memoria.
75          add edx,2                   ; Apuntar a la siguiente palabra.
76          loop ciclo_llenado          ; LOOP siempre usa CX como contador.
77          mov eax,cr0                 ; Obtener el valor del registro de control 0.
78          and al,0FEh                 ; Poner a cero el bit Protection Enable (PE).
79          mov cr0,eax                 ; Cargarlo. En este momento el uP esta en
80                                      ; modo real nuevamente.
81          jmp short $+2               ; Vaciar la cola de instrucciones.
82          sti                         ; Volver a habilitar interrupciones.
83          mov ax,4C00h                ; Funcion de DOS para terminar el programa.
84          int 21h
85 codigo   ends                        ; Final del unico segmento.
86          end comienzo
```

```
; Hacer un programa que muestre la cantidad de decimas de segundo que estuvo
; corriendo en modo protegido. Para salir al modo real el usuario podra
; apretar cualquier tecla. Utilizar para ello las interrupciones en modo
; protegido 8 (reloj) y 9 (teclado).
;
; Hecho por Dario Alejandro Alpern el 9 de setiembre de 1999.
;
   .386p
nada    segment use16 at 0
dummy   label far
nada    ends
codigo  segment use16
   assume cs:codigo,ds:codigo
   org 100h
comienzo:jmp inicio
; Valores que van en la tabla de descriptores globales (dos descriptores)
; 1er descriptor: Segmento de codigo.
gdtab   db 0FFh         ;Bits 7-0 del limite.
        db 0FFh         ;Bits 15-8 del limite.
        db 0            ;Bits 7-0 de la base (se llena durante la ejecucion).
        db 0            ;Bits 15-8 de la base (se llena durante la ejecucion).
        db 0            ;Bits 23-16 de la base (se llena durante la ejecucion).
        db 9Ah          ;Byte de derechos de acceso:
                          ;Bit 7=1: Segmento Presente.
                          ;Bits 6,5=00: Nivel de Privilegio cero.
                          ;Bit 4=1: Segmento de codigo o datos.
                          ;Bit 3=1: Descriptor correspondiente a codigo.
                          ;Bit 2=0: Segmento no conforme.
                          ;Bit 1=1: El segmento de codigo se puede leer.
                          ;Bit 0=0: El segmento no fue accedido.
        db 0            ;Bit 7=0: Granularidad = 1 byte.
                        ;Bit 6=0: Segmento de 16 bits.
                        ;Bit 5,4=00: No usados.
                        ;Bits 3-0=0000: Bits 19-16 del limite.
        db 0            ;Bits 31-24 de la base.
; 2do descriptor: Segmento de datos.
        db 0FFh         ;Bits 7-0 del limite.
        db 0FFh         ;Bits 15-8 del limite.
        db 0            ;Bits 7-0 de la base.
        db 0            ;Bits 15-8 de la base.
        db 0            ;Bits 23-16 de la base.
        db 92h          ;Byte de derechos de acceso:
                          ;Bit 7=1: Segmento Presente.
                          ;Bits 6,5=00: Nivel de Privilegio cero.
                          ;Bit 4=1: Segmento de codigo o datos.
                          ;Bit 3=0: Descriptor correspondiente a datos.
                          ;Bit 2=0: Offset <= Limite.
                          ;Bit 1=1: El segmento de datos se puede escribir.
                          ;Bit 0=0: El segmento no fue accedido.
        db 0Fh          ;Bit 7=0: Granularidad = 1 byte.
                        ;Bit 6=0: Segmento de 16 bits.
                        ;Bit 5,4=00: No usados.
                        ;Bits 3-0=1111: Bits 19-16 del limite.
        db 0            ;Bits 31-24 de la base.
; Valores que van en la tabla de descriptores de interrupcion
; (dos descriptores correspondientes a INT 8 e INT 9)
; 1er Descriptor: Compuerta de interrupcion correspondiente a INT 8.
idtab   dw int8han      ;Bits 15-0 del offset.
        dw cs_sel       ;Selector del segmento de codigo.
        db 0            ;Cantidad de palabras que ocupan los parametros.
        db 86h          ;Indica que es una compuerta de interrupcion tipo 286.
        dw 0            ;Bits 31-16 del offset.
; 2do Descriptor: Compuerta de interrupcion correspondiente a INT 9.
        dw int9han      ;Bits 15-0 del offset.
        dw cs_sel       ;Selector del segmento de codigo.
```

```
 66         db 0            ;Cantidad de palabras que ocupan los parametros.
 67         db 86h          ;Indica que es una compuerta de interrupcion tipo 286.
 68         dw 0            ;Bits 31-16 del offset.
 69 cs_sel  equ 8           ;Valor del selector del segmento de codigo.
 70 ds_sel  equ 16          ;Valor del selector del segmento de datos.
 71 gdt     equ gdtab - 8   ;Inicio de la tabla de descriptores globales.
 72 idt     equ idtab - 64  ;Inicio de la tabla de interrupcion.
 73 texto1  db "Este programa no puede correr en modo virtual 8086.",13,10
 74         db "Pruebe eliminando la linea DEVICE=EMM386 en CONFIG.SYS",13,10,"$"
 75 texto2  db "Este programa requiere 80386 o posterior.",13,10,"$"
 76 texto3  dw 24*160+25*2
 77         db "Hecho por Dario Alpern el 9/9/1995.",0
 78 texto4  dw 11*160+25*2
 79         db "Segundos en modo protegido:",0
 80 texto5  dw 13*160+25*2
 81         db "Apriete cualquier tecla para salir.",0
 82 picint  db 0
 83 gdtr    label fword     ;Informacion a almacenar en el GDTR.
 84         dw 3*8-1        ;Limite de la tabla de descriptores globales.
 85         dd 0            ;Base de la tabla de descriptores globales.
 86                         ;Se llena durante la ejecucion del programa.
 87 idtr    label fword     ;Informacion a almacenar en el IDTR.
 88         dw 10*8-1       ;Limite de la tabla de descriptores de interrupcion.
 89         dd 0            ;Base de la tabla de descriptores de interrupcion.
 90 real_idtr df 0            ;Espacio para almacenar el IDTR en modo real.
 91 final   db 0            ;Indicador para saber si se apreto una tecla.
 92 tics    dd 0            ;Tiempo (en tics de reloj) desde que comenzo el prog.
 93 inicio: mov dx,offset texto2
 94 test86: pushf           ;Sirve para rechazar un 8086. En este procesador
 95         pop ax          ;los bits 15-12 del registro de indicadores siempre
 96         and ax,0FFFh    ;estan a uno.
 97         push ax
 98         popf
 99         pushf
100         pop ax
101         add ax,1000h
102         jc short mal
103 test286:pushf           ;Sirve para rechazar un 80286. En este procesador
104         pop ax          ;los bits 15-12 del registro de indicadores siempre
105         or ax,0F000h    ;estan a cero en modo real.
106         push ax
107         popf
108         pushf
109         pop ax
110         and ax,0F000h
111         jz short mal
112 virtual?:smsw ax        ;Sirve para ver si el 80386 esta en modo virtual.
113         test al,1       ;Aqui no puede utilizarse MOV EAX,CR0.
114         jz short modo_real
115         mov dx,offset texto1
116 mal:    mov ah,9        ;Mostrar el mensaje de error.
117         int 21h
118         mov ax,4c01h    ;Terminar el programa con codigo de salida 1.
119         int 21h
120 modo_real:in al,21h     ;Almacenar en un lugar temporal los IRQ habilitados
121         mov picint,al   ;en el controlador de interrupciones (PIC).
122         mov real_cs,cs  ;Llenar el campo correspondiente al segmento en el
123                         ;salto intersegmento que esta mas abajo.
124         mov ah,0fh      ;Averiguar el modo de video segun la BIOS. Si vale 7,
125         int 10h         ;el buffer de video comienza en B0000h, en caso
126         mov ebp,0B8000h ;contrario, en B8000h.
127         cmp al,7
128         jnz short ebp_ok
129         xor bp,bp
130 ebp_ok: mov ax,cs       ;Llenar la base del descriptor del segmento de codigo
```

```
131         xor dx,dx       ;en la tabla de descriptores globales con la misma
132         shld dx,ax,4    ;base que para el modo real.
133         shl ax,4        ;DL:AX = Direccion lineal del segmento de codigo.
134         mov byte ptr gdt[cs_sel+4],dl
135         mov word ptr gdt[cs_sel+2],ax
136         mov bx,ax
137         mov cl,dl
138         add ax,offset gdt
139         adc dl,0        ;DL:AX = Direccion lineal de la GDT.
140         mov word ptr gdtr[2],ax
141         mov byte ptr gdtr[4],dl
142         add bx,offset idt
143         adc cl,0        ;CL:BX = Direccion lineal de la IDT.
144         mov word ptr idtr[2],bx
145         mov byte ptr idtr[4],cl
146         mov al,0FFh     ;Deshabilitar todas las IRQ del PIC.
147         out 21h,al
148         jmp $+2
149         sidt real_idtr  ;Almacenar en memoria el IDTR del modo real.
150         lgdt gdtr       ;Cargar el GDTR.
151         lidt idtr       ;Cargar el IDTR.
152         push cs_sel     ;Poner en la pila el descriptor y el offset del
153         push offset modo_protegido ;codigo en modo protegido.
154         mov eax,cr0     ;Pasar a modo protegido poniendo a uno el bit 0
155         or al,1         ;(Protection Enable) de CR0.
156         mov cr0,eax
157         retf            ;Ir a ejecutar codigo en modo protegido.
158                         ;En este caso es la instruccion siguiente.
159 modo_protegido:
160         mov ax,ds_sel
161         mov ds,ax       ;Poner en DS el selector del segmento de datos.
162         mov ebx,ebp     ;Limpiar la pantalla. Notese que el offset supera
163         mov cx,2000     ;los 64KB en modo protegido (con la eleccion
164         mov ax,0720h    ;realizada en la GDT debera ser menor que 1MB).
165 bucle_cls:mov [ebx],ax
166         add ebx,2
167         loop bucle_cls
168         mov si,offset texto3 ;Mostrar las tres leyendas en pantalla.
169         call mostrar_texto
170         mov si,offset texto4
171         call mostrar_texto
172         mov si,offset texto5
173         call mostrar_texto
174         mov al,0FCh     ;Habilitar unicamente la INT 8 (reloj) e INT 9
175         out 21h,al      ;(teclado).
176         jmp $+2
177 espera: cmp ss:final,0  ;Se apreto una tecla?
178         jz espera       ;Saltar si no es asi.
179         mov al,0FFh     ;Deshabilitar todas las interrupciones.
180         out 21h,al
181         jmp $+2
182         mov eax,cr0     ;Volver a modo real poniendo a cero el bit 0 de CR0.
183         and al,0feh
184         mov cr0,eax
185         jmp far ptr dummy ;Aqui es absolutamente necesario un salto DIRECTO
186                           ;intersegmento. No funciona ni el metodo de RETF
187                           ;(como arriba) ni un salto indirecto.
188         org $-4
189         dw regreso_a_modo_real ;Offset del salto.
190 real_cs dw 0              ;Segmento del salto.
191 regreso_a_modo_real:
192         lidt ss:real_idtr ;Restaurar el valor de IDTR.
193         mov al,ss:picint  ;Restaurar las habilitaciones de la PIC.
194         out 21h,al
195         jmp $+2
```

```
196         mov ax,4c00h    ;Terminar el programa.
197         int 21h
198 ;La siguiente subrutina muestra un texto ASCIIZ apuntado por ss:[si].
199 mostrar_texto:
200         movzx ebx,word ptr ss:[si]
201         add ebx,ebp     ;EBX = Direccion donde debe ir el texto en pantalla.
202         inc si
203 bucle_disp:inc si       ;Apuntar al siguiente caracter.
204         mov al,ss:[si]  ;Obtener el proximo caracter a mostrar.
205         and al,al       ;Ver si se termino.
206         jz short fin_disp ;Saltar si es asi.
207         mov [ebx],al    ;Mandarlo a pantalla.
208         add ebx,2       ;Apuntar a la siguiente posicion en pantalla.
209         jmp bucle_disp
210 fin_disp:ret               ;Terminar la subrutina.
211 ;Manejador de la interrupcion de reloj.
212 int8han:pushad             ;Preservar todos los registros de 32 bits de uso gral.
213         mov eax,ds:[046Ch] ;Actualizar el tic de reloj para mantenerlo en hora.
214         inc eax
215         cmp eax,180040h ;Llego medianoche?
216         jnz short guardar_tic   ;Saltar si no es asi.
217         inc byte ptr ds:[0470h] ;Indicarlo en la variable de la RAM BIOS.
218         xor eax,eax
219 guardar_tic:mov ds:[046Ch],eax
220         inc dword ptr ss:tics   ;Incrementar tics desde inicio del programa.
221         mov eax,35997*65536     ;Convertir a decimos de segundo.
222         mul dword ptr ss:tics
223         mov ebx,ebp             ;Obtener posicion a mostrar en pantalla.
224         add bx,ss:texto4
225         add bx,(texto5 - texto4) * 2 ;EBX = Posicion en pantalla.
226         mov cl,0
227         mov eax,1000000000
228         call digito
229         mov eax,100000000
230         call digito
231         mov eax,10000000
232         call digito
233         mov eax,1000000
234         call digito
235         mov eax,100000
236         call digito
237         mov eax,10000
238         call digito
239         mov ax,1000
240         call digito
241         mov ax,100
242         call digito
243         mov cl,1
244         mov al,10
245         call digito
246         mov ch,"."
247         call mostr_dig
248         mov al,1
249         call digito
250 fin_inter:mov al,20h         ;Indicarle al PIC que finalizo la interrupcion.
251         out 20h,al
252         popad               ;Restaurar los registros.
253         iret                ;Fin de la interrupcion.
254 digito: mov ch,"0"-1
255 bucle_dig:inc ch
256         sub edx,eax
257         jnc bucle_dig
258         add edx,eax
259         and cl,cl
260         jnz short mostr_dig
```

```
261         cmp ch,"0"
262         jz short fin_digito
263         mov cl,1
264 mostr_dig:mov [ebx],ch
265         add ebx,2
266 fin_digito:ret
267 int9han:pushad                  ;Preservar los registros de 32 bits.
268         in al,60h               ;Obtener informacion del controlador de teclado
269         and al,al               ;La tecla se acaba de apretar o de soltar?
270         js fin_inter            ;Saltar si se acaba de soltar.
271         mov ss:final,1          ;Indicar que se termine el programa.
272         jmp fin_inter           ;Ir a restaurar los registros.
273 codigo  ends
274         end comienzo
```