Ejercicio TC75-02p

Cierto periférico conectado al bus de expansión de un sistema PC usa cuatro puertos de E/S consecutivos, seleccionables mediante conmutadores al instalar la tarjeta; sea BASE la dirección del primer puerto. Se numeran los puertos del 0 al 3 y se suma este número a la dirección BASE. El puerto 0 es el byte menos significativo (LSB) del puerto de datos. El puerto 1 es el byte más significativo. El puerto 2, de entrada, es el puerto de estado y el mismo puerto 2, de salida es el puerto de control. El puerto 3 se decodifica pero no se usa.
El puerto de control, de sólo escritura, es un registro de 8 bits que controla diversas funciones del periférico. Tras el encendido del sistema todos los bits del registro de control están a cero.

Registro de control:

7	6	5	4	3	2	1	0
A2	HLD	X	ENINT	RESET	A1	X	ENDMA

Bit 0: ENDMA, si = 1 habilita al DMA, es decir que el periférico escribe y lee la memoria a través del canal DMA5, de 16 bits.
Bit 3: RESET, si = 1 al periférico se le permite funcionar normalmente; si = 0 el periférico entra en estado de Reset.
Bit 4: ENINT, si = 1 se habilita el periférico para interrumpir al PC activando IRQ10 cuando ocurran DDR ó SDR.
Bit 6: HLD, si = 1 se manda al periférico que entre en estado "suspendido" o latente; si = 0 se manda al periférico que salga del estado "suspendido". El periférico entra y sale del estado suspendido cuando puede, no inmediatamente.
Bits 2 y 7: tienen otras funciones que no interesan aquí.
Bits 1 y 5: no se usan.
El puerto de estado, de sólo lectura, comunica al PC el estado actual del periférico.

Registro de estado:

7	6	5	4	3	2	1	0
X	X	X	X	DDR	HLDA	X	SDR

Sólo emplea tres bits.
Bit 0: SDR (*System Data Ready*), si = 1 indica que el PC tiene datos listos para enviar al periférico. Se desactiva automáticamente cuando el periférico lee el puerto de datos.
Bit 2: HLDA (*HoLD Acknowledge*), si = 0 indica que el periférico está en estado suspendido; si = 1 indica que el periférico está corriendo normalmente.
Bit 3: DDR (*Device Data Ready*), si = 1 indica que el periférico tiene datos listos para enviar al PC. Se desactiva automáticamente cuando el PC lee el puerto de datos.
Los puertos 0 y 1 forman un puerto de 16 bits bidireccional a través del cual se pasan datos (WORDS) entre el PC y el periférico. La comunicación está protocolizada por las señales DDR y SDR.
Escribir, en lenguaje ensamblador del 80x86, las rutinas necesarias para:

a) Poner en modo Reset al periférico y sacarlo de ese estado.
b) Escribir una WORD de datos al periférico, comprobando antes que éste no está en modo suspendido (pues el periférico no "oiría" al PC).
c) Suspender al periférico y comprobar que está suspendido.

Solución

a) El estado `RESET` del periférico se controla con el bit 3 del registro de control. Por tanto, cambiando este bit se controla dicho estado. Ahora bien, como el registro de control es de sólo escritura para escribir en él no podemos saber qué valor tiene de antemano. Para ello lo que se hace es mantener una copia de este registro en la memoria RAM. Así encerramos los accesos al registro de control de la siguiente manera:

```
.DATA
Reg_Control     BYTE    ?

.CODE
WriteControl    MACRO   Modo
        mov     al,Modo
        mov     dx,BASE+2
        mov     Reg_Control,al
        out     dx,al
        ENDM

ReadControl     MACRO
        mov     al,Reg_Control
        ENDM
```

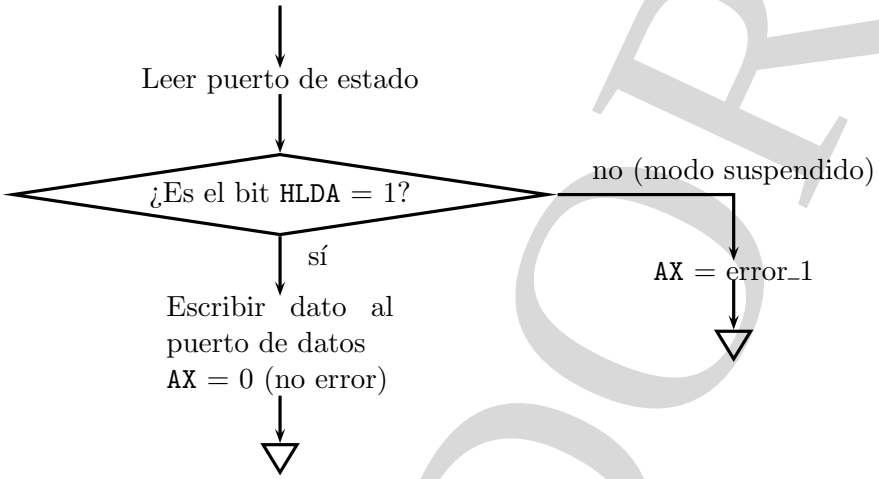
Ahora para modificar exclusivamente a algún bit del registro de control se hace así:

```
ResetDevice     MACRO   Modo
        and     Reg_Control,0F7h        ;pone a 0 el bit 3
        WriteControl    Reg_Control
        ENDM

SetDevice       MACRO
        or      Reg_Control,08h
        WriteControl    Reg_Control
        ENDM
```

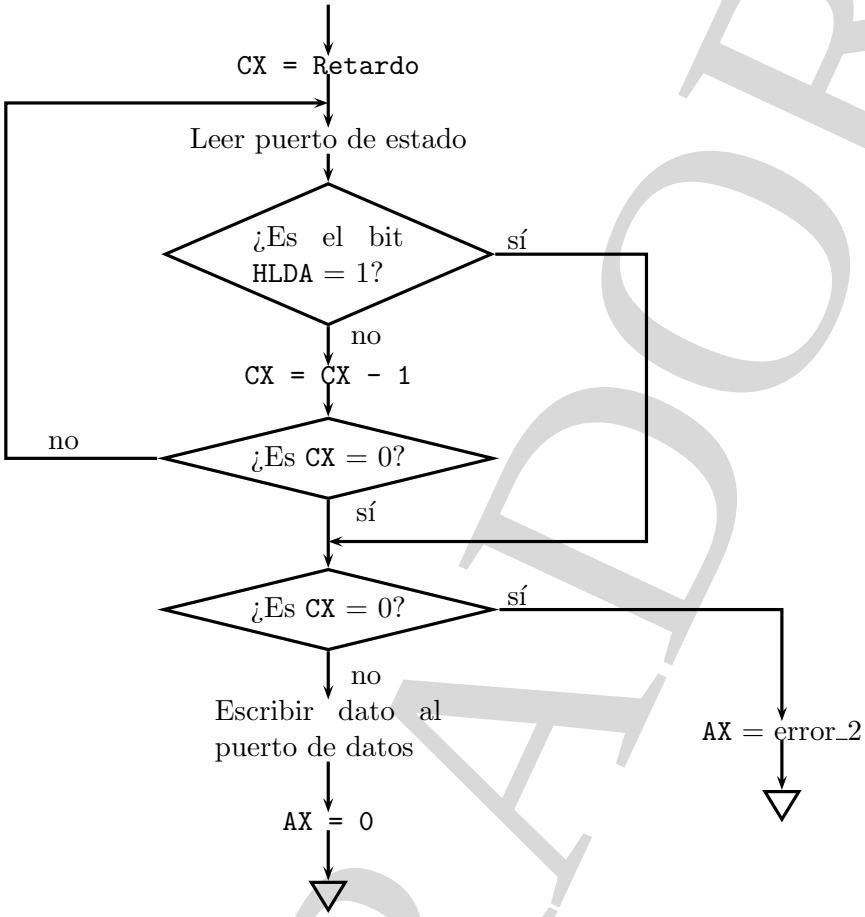
b) Para pasarle una palabra de datos hay que comprobar previamente que no esté en estado "suspendido". Hay dos alternativas: la primera es comprobar si el periférico está suspendido o no; si está suspendido se devuelve un indicador de error y se sale; si no lo está se le pasa la palabra de datos. La otra alternativa es interrogar contínuamente al periférico (bucle *tábano*) hasta que éste salga del estado suspendido y entonces pasarle el dato; para no quedarse "colgado" permanentemente, como el periférico, se le da un tiempo prudencial y si se cumple se devuelve un código de error. La segunda alternativa es más compleja y en ciertas situaciones tiene ventajas.

*** Método 1:**




```
WriteData       MACRO   Dato
        LOCAL   @@local,@@final
        mov     dx,BASE+2
        in      al,dx
        test    al,04h
        jnz     @@local
        mov     ax,error_1
        jmp     @@final
@@local:
        mov     dx,BASE
        mov     ax,Dato
        out     dx,ax
        xor     ax,ax
@@final:
        ENDM
```

*** Método 2:** El bucle de retardo se controla con una variable ´´`Retardo`"




```
WriteData       MACRO   Dato
        LOCAL   @@bucle,@@final
        mov     cx,Retardo
        mov     dx,BASE+2
@@bucle:
        in      al,dx
        test    al,04h      ;;si HLDA=0, resultado=0, ZF=1
        loopz   @@bucle
        mov     ax,error_2
        jcxz    @@final
        mov     dx,BASE
        mov     ax,Dato
        out     dx,ax
        xor     ax,ax
@@final:
        ENDM
```

c) Suspender el periférico: hay que poner a 1 el bit 6 del registro de control. A continuación hay que leer contínuamente (*gadfly loop*) el bit `HLDA` del registro de estado hasta que se ponga a 0.

```
HoldDevice      MACRO   Modo
```

```
        or      Reg_Control,40h
        WriteControl    Reg_Control
@_1:
        in      al,dx
        test    al,04h
        jnz     @_1
        ENDM
```