GENERACIÓN* DE *SONIDOS* DE DIFERENTES *FRECUENCIAS

Esta práctica está centrada a los ***sonidos*** que se pueden emitir en una computadora a través de un programa en ***LENGUAJE ENSAMBLADOR***. Todos los ***sonidos*** que se emiten, se pueden variar mediante la ***frecuencia***, la cual está dada en veces por segundo.

1.0 ***Capturar*** el siguiente módulo y guardarlo en **SONIDO.ASM**.

```
%TITLE "RUTINAS PARA PRODUCIR TONOS"
IDEAL
DOSSEG
MODEL small
DATASEG
ContReloj DW ?,?
ContA DW 1000 ;Para nota
ContD DW 1000 ;Para silencio
Entera DW 2000 ;Nota entera
Notas DW 4186 ;C
DW 4435
DW 4699 ;D
DW 4978
DW 5274 ;E
DW 5588 ;FDW 5920
DW 6272 ;G
DW 6645
DW 7040 ;A
DW 7459
DW 790 ;B
CODESEG
PUBLIC Toca
```

```
////////////////////////////////////////////////////////////////////////////////
;Frec CALCULA CONSTANTE PAR FIJAR FRECUENCIA
;
;Entrada: CX =frecuencia en Hertz
;Salida: CX = Cociente a enviar al puerto 42H
;Registros: CX
////////////////////////////////////////////////////////////////////////////////

PROC Frec
push dx ;Salva registros
push ax
mov dx,12h ;Parte superior del numerador
mov ax,34deh ;Parte inferior del numerador
div cx ;Divide entre frecuencia
mov cx,ax ;El cociente es la salida
pop ax ;Repone registros
pop dx
ENDP Frec
```

Salvamos registros, hacemos la división con el registro **CX**. El resultado está en el registro

AX, lo movemos al registro **CX**, ya que el registro **CX** tendrá la frecuencia usada.

2.0 ***Capturar*** el módulo siguiente y **salvarlo** en **SONIDO.ASM**

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

;PoneTono FIJA LA FRECUENCIA EN LA BOCINA f = F/n. F = 1.193,182 Hz
;
;Entrada: CX = constante para el temporizador
;Salida: Registros de 8253 quedan con la constante
;Registros: Ninguno
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

PROC PoneTono
push ax
```

```
;Carga período en el timer
mov al,cl ;Byte bajo, primero
out 42h,al ;Envía al 8254
mov al,ch ;Byte alto, después
out 42h,al
pop ax ;Repone registro
ENDP PoneTono
```

El registro **CX** contiene la frecuencia. Movemos la parte baja del registro **CX** al registro **AL**
para mandar el contenido de **AL** a la puerta de salida **42h**.
La parte alta del registro **CX** la mandamos al registro **AL** y lo ponemos en la puerta de
salida **42h**.

3.0 ***Capturar*** el módulo siguiente y **salvarlo** en **SONIDO.ASM**

```
////////////////////////////////////////////////////////////////////////////////////////////////////

;ATono ARRANCA (INICIA) EL TONO EN EL ALTAVOZ
;
;Entrada: Ninguna
;Salida: Tono audible
;Registros: Ninguno
////////////////////////////////////////////////////////////////////////////////////////////////////

PROC ATono
push ax ;Salva registro
in al,61h ;Trae contenido de puerto B
or al,03 ;Enciende bocina y timer
out 61h,al ;Saca nuevo valor de puerto B
pop ax ;Repone registro
ENDP ATono
```

Traemos byte al puerto de entrada, posteriormente encendemos el altavoz y sacamos el
nuevo valor del registro **AL** por el puerto de salida **61h**.

4.0 ***Salvar*** el módulo siguiente en **SONIDO.ASM**

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
;Lapso TARDA UN NÚMERO DADO DE MILISEGUNDOS
;
;Entrada: CX = milisegundos
;Salida: Ninguna
;Registros: CX
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
PROC Lapso
push ax dx cx bx ;Salva registros
mov dx,0
mov ax,cx
mov bx,55 ;18.2 tics/seg
div bx
mov bx,ax ;BX = número de tics
mov ah,00 ;Trae tics del día con BIOS
int 1ah
mov [word ContReloj],cx ;Salva parte alta
mov [word ContReloj + 2],dx ;Salva parte baja
@@10:
mov ah,0 ;Lee con BIOS los tics
int 1ah
sub dx,bx ;Les resta tics del lapso
sbb cx,0
cmp cx,[word ContReloj] ;Compara parte alta
jb @@10 ;Si es menor vuelve a leer
ja @@20 ;Si mayor, ya termino
cmp dx,[word ContReloj+2] ;Iguales, compara parte inferior
```

```
jb @@10
@@20:
pop bx cx dx ax ;recobra registros
ret
ENDP Lapso
```

Salvamos registros. Cargamos registro **DX** con cero, registro **AX** con el tiempo del registro **CX** y el registro **BX** con **55**. Dividimos el registro **BX**, el resultado se encuentra en **AX**, lo colocamos en el registro **BX** (número de tics).
Usamos el servicio **0** de la interrupción **1Ah** para obtener los tics del reloj y salvarlos en la variable: **ContReloj dw ?,? (CX,DX)** Volvemos a usar la interrupción **1Ah** para obtener los nuevos tics del reloj.
Restamos el registro **DX** con el número de tics (**BX**). El resultado es almacenado en **DX**.
Hacemos una resta con préstamo del registro **CX** con cero.
Comparamos la parte alta de la variable ***ContReloj*** con el registro **CX**; si es menor salta a **@@10** para volver a usar la interrupción **1Ah**; si es mayor termina y recupera registros; si no suceden ambos casos comparamos la parte inferior de la variable ***ContReloj*** con el registro **DX**; si es menor salta a **@@10** para usar la interrupción **1Ah**; si no termina con la subrutina y recupera registros.

5.0 ***Salvar*** el módulo siguiente en **SONIDO.ASM**

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
;QTono QUITA (APAGA) EL TONO
;
;Entrada: Ninguna
;Salida: Ninguna (Tono deja de oírse)
;Registros: Ninguno
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
PROC QTono
push ax ;Salva registro AX
in al,61h ;Trae valor de puerto B
and al,0fch ;Apaga altavoz y timer
```

```
out 61h,al
pop ax ;Recupera registro AX
ENDP QTono
```

Traemos valor a puerta de entrada, apagamos el altavoz y sacamos el valor del registro

AL

por la puerta de salida **61h**.

6.0 ***Capturar*** los módulos siguientes en **SONIDO.ASM**

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
;Nota CONVIERTE EL VALOR DE NOTA (0 A 95) A FRECUENCIA
;
;Entrada: AL = número de escala cromática extendida
;Salida: DX = valor adecuado para PoneTono
;Registros: Ninguno
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
PROC Nota
push cx ;Salva registros que
push bx ;usa
push ax
mov ah,0 ;Extiende número de nota
mov cl,12 ;Divide entre 12
div cl ;Has la división
mov dl,al ;El cociente da la octava
mov al,ah ;El residuo es el índice
cbw ;Búsqueda requiere 16 bits
shl ax,1 ;Dos bytes por nota
mov bx,ax ;Usa direccionamiento base
mov cx,[notas + bx] ;Trae de la tabla
call Frec ;Convierte la frecuencia
xchg cx,dx ;Octava en CL, periodo en DX
neg cl ;Contador de corrimiento
add cl,8 ;= 8 - octava
sal dx,cl
```

```
pop ax ;Recupera registros
pop bx
pop cx
ENDP Nota
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
;Toca RUTINA QUE TOCA MÚSICA A PARTIR DE DATOS BINARIOS
;
;Entrada: DS: SI apunta a lista de datos binarios, formada por 4 comandos:
; T, N, D y X, que forman la tonada, donde:
;
; Comando Tiempo:
; 1er. Byte = ASCII 'T'
; 2do. Byte = tiempo en notas enteras por minuto
;
; Comando Nota:
; ; 1er. Byte = ASCII 'N'
; 2do. Byte = número de nota (0 a 95)
; 3er. Byte = largo (binario de punto fijo 8 bits, escala 1)
; 4to. Byte = estilo (binario de punto fijo 8 bits, escala 0)
;
; Comando Descanso:
; 1er. Byte = ASCII 'D''
; 2do. Byte = largo (binario de punto fijo 8 bits, escala 1)
;
; Comando de Terminación:
; 1er. Byte = ASCII 'X'
;
;Salida: A bocina y temporizador solamente
;Registros: Ninguno'

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
PROC Toca
push si ;Salva registros usados
push dx
```

```
push cx
push bx
push ax
; pone tempo por omisión..
mov [Entera],2000 ;2,000 ms para una nota entera
cld ;Incrementando
@@10:
lodsb ;Trae byte de la lista
; checa si llega comando de terminación..
cmp al,'X' ;¿Final ?
jne @@20
jmp @@99
; comando tempo.
@@20:
cmp al,'T' ;¿Comando tiempo?
jne @@30 ;No, brinca
lodsb ;Trae tempo
mov cl,al ;Lo pone en CX
mov ch,0
mov ax,60000 ;Milisegundos en un minuto
mov dx,0 ;Borra parte superior
div cx ;Divide entre el tiempo
mov [Entera],ax ;ms por nota entera
jmp @@10
; comando nota..
@@30:
cmp al,'N' ;¿Comando nota ?
jne @@40 ;No, brinca
lodsb ;Trae número de nota
call Nota ;Convierte
mov cx,dx ;Resultado en CX
call PoneTono ;Pone la frecuencia
call Atono ;Prende el altavoz
mov cx,[Entera] ;Trae ms por nota entera
```

```
lodsb ;Trae duración
mov ah,al ;Prepara multiplicador
mov al,0 ;de duración
sal cx,1 ;Factor de escala 1
mul cx ;Multiplica
mov cx,dx ;Contador total para la nota
lodsb ;Trae estilo
mov ah,al ;Prepara multiplicador
mov al,0 ;de estilo
mul cx ;Multiplica
mov [ContA],dx ;Salva contador para nota
sub cx,dx ;Contador para descanso
mov [ContD],cx ;Lo salva
mov cx,[ContA] ;Parte audible de la nota
call Lapso ;Retardo
call QTono ;Apaga altavoz
mov cx,[contD] ;Parte inaudible de la nota
call Lapso
jmp @@10
; Comando de descanso..
@@40:
cmp al,'D' ;¿Comando de silencio ?
jne @@99 ;No, salir
mov cx,[Entera] ;Trae ms por nota entera
lodsb ;Trae duración
mov ah,al ;Prepara multiplicador
mov al,0 ;de duración
sal cx,1 ;Factor para escala 1
mul cx ;Multiplica
mov cx,dx ;Contador total
call Lapso
jmp @@10
@@99:
pop ax ;Recupera
```

```
pop bx
pop cx
pop dx
pop si
ret
ENDP Toca
END ;Fin de módulo SONIDO.ASM.
```

Salvamos registros. Colocamos 2,000 milisegundos en variable [Entera].
Traemos Byte de
la lista y lo comparamos con "**X**", con "**T**", con "**N**" y con "D".

6.0 ***Ensamblar*** completamente el programa **SONIDO.ASM**, mediante el comando:

C>tasm/zi sonido

8.0 ***Meter*** en librería el módulo **SONIDO.ASM**, mediante el comando:

C>tlib rem -+sonido

9.0 ***Capturar*** el siguiente programa con el nombre de **cuca.ASM**.

```
; cucaracha
%TITLE 'EMITE POR EL ALTAVOZ UN FRAGMENTO DE CONCIERTO'
IDEAL
DOSSEG
MODEL small
STACK 512
DATASEG
ClaveFin DB 0
Himno DB 'T',18
DB "N",76,35,128
DB 'D',56
```

```
DB "N",76,35,128
DB 'D',56
DB "N",76,35,128
;DB 'D',64
DB "N",69,250,128
DB "N",73,250,128
DB 'D',56
DB "N",76,35,128
DB 'D',56
DB "N",76,35,128
DB 'D',56
DB "N",76,35,128
;DB 'D',64
DB "N",69,250,128
DB "N",73,250,128
DB 'N',69,20,128
DB 'D',56
DB 'N',69,20,128
DB 'D',56
DB 'N',68,35,128
DB 'D',56
DB 'N',68,35,128
DB 'D',56
DB 'N',66,65,228
DB 'D',56
DB 'N',66,65,228
DB 'D',24
DB 'N',64,35,128
DB 'D',56
DB "N",76,35,128
DB 'D',56
DB "N",76,35,128
DB 'D',56
DB "N",76,35,128
```

```
DB 'D',56
DB 'N',68,250,128
DB 'N',71,250,128
DB "N",76,35,128
DB 'D',56
DB "N",76,35,128
DB 'D',56
DB "N",76,35,128
DB 'D',56
DB 'N',68,250,200
DB 'N',71,250,228
DB 'D',56
DB 'N',76,65,128
DB 'D',112
DB 'N',78,65,128
DB 'D',112
DB 'N',76,65,128
DB 'D',112
DB 'N',86,65,128
DB 'D',192
DB 'N',85,95,128
DB 'D',192
DB 'N',83,95,128
DB 'D',192
DB 'N',81,130,128
;DB 'N',76,35,128
DB 'X'
CODESEG
; De SONIDO.OBJ
EXTRN Toca:proc
Inicia:
mov ax,@data ;Inicia segmento de datos
mov ds,ax
mov es,ax
```

```
mov si,offset Himno ;Apunta a tabla
call Toca ;Ejecuta melodía
mov ah,4ch ;Regresa a DOS
mov al,[ClaveFin]
int 21h
END Inicia ;
```

10.0 ***Ensamblar*** el programa anterior y **hacerlo** ejecutable.

C:\>tasm /zi cuca

C:\>tlink /v cuca,,,rem

11.0 ***Ejecutar*** el programa:

C:\>cuca