Escuela Politécnica Superior

UAM
Universidad Autónoma de Madrid

Laboratorio de Sistemas Basados en Microprocesadores

Práctica 2: Juego de instrucciones.

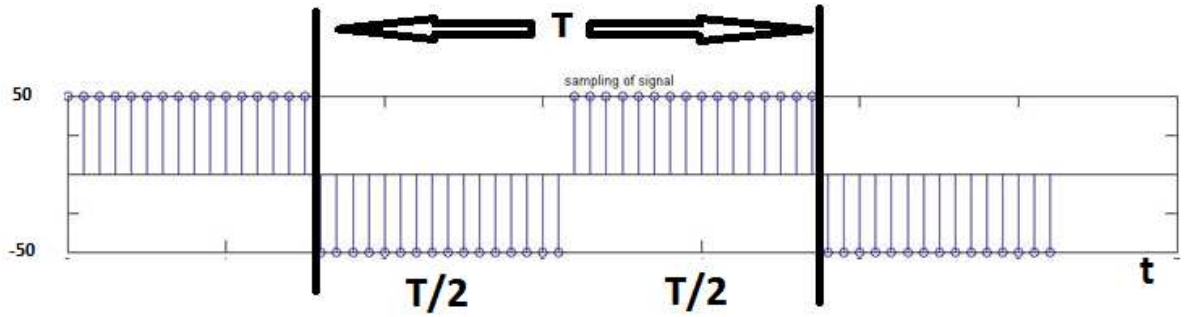
Esta práctica consiste en el desarrollo de programas sencillos para generar, escribir, leer y procesar señales de sonido monoaural almacenadas en disco mediante archivos codificados en formato WAV. En cada ejercicio, se deberá implementar un programa principal que llame a funciones públicas lejanas de un módulo externo que se proporciona con el material de la práctica (*wav.obj*). Este módulo encapsula las funciones básicas de lectura y escritura de ficheros WAV especificadas en el anexo de este enunciado. Dicho módulo deberá enlazarse con el código objeto del programa principal.

El formato WAV

Debido a su capacidad de compresión, el formato MP3 se ha convertido en el más utilizado para almacenar audio. Sin embargo, debido a su simplicidad, el formato WAV resulta el más conveniente para hacer ejercicios de procesamiento digital de señales.

Los archivos WAV se componen de varias secciones y pueden contener el audio en diversas formas. En el anexo de la práctica, se incluye una descripción de los diferentes campos incluidos en el formato WAV.

En todos los ejercicios, las señales de sonido mono serán ondas cuadradas de un segundo de duración con amplitudes de +50 y -50. La frecuencia de muestreo será de 8000 muestras por segundo (8 KHz). Cada muestra estará codificada mediante un byte con signo en complemento a dos. Por tanto, cada señal cuadrada consistirá de una secuencia de 8000 bytes.



El alumno deberá realizar el correspondiente análisis de los requisitos expuestos e implementar el código fuente convenientemente comentado. Para el desarrollo de esta práctica se usará una buena parte de instrucciones básicas del 80x86.

Para la verificación de los ficheros WAV, se recomienda traer **auriculares** con conexión "jack" de audio de 3,5mm para conectar a la tarjeta de audio de los ordenadores de la Escuela.

Para el desarrollo de los ejercicios de esta práctica, se dispondrá de las siguientes rutinas contenidas dentro del módulo *wab.obj*:

- ***Init_WAV_Header***: Esta rutina inicializa la cabecera necesaria para los ficheros WAV.
- ***fopen*, *fclose*, *fread***: Rutinas necesarias para la gestión de ficheros en disco: apertura, cierre y lectura respectivamente.
- ***Write_WAV*:** Escribir una tabla de bytes de audio de memoria a un fichero en formato WAV.
- ***Read_WAV*:** Leer en memoria una tabla de bytes de audio desde un fichero en formato WAV.

Además de estos procedimientos, **será necesario utilizar las rutinas incluidas** en la int 21h para la impresión en pantalla y acceso desde el teclado. Todas estas rutinas se encuentran detalladas en los anexos.

Programa pract2a.asm (5 puntos)

Desarrollar un programa que genere una señal cuadrada de 440 Hz de frecuencia (nota La / A4) y la almacene en disco en un archivo WAV con el nombre "*gen_la.wav*".

El programa efectuará las siguientes tareas:

- Inicializar el módulo WAV especificando una frecuencia de muestreo (*sample rate*) de 8000 muestras por segundo y un número de muestras de 8000 (un segundo de duración). Se invocará a la función lejana *Init_WAV_Header* del módulo.
- Generar en una tabla almacenada en memoria principal una señal cuadrada de 440 Hz de frecuencia. Esto significa que los 8000 bytes que codifican la señal deberán representar una onda que oscile 440 veces entre los valores +50 y -50.
- Abrir un fichero en disco con el nombre "*gen_la.wav*". Se invocará a la función lejana *fopen* del módulo.
- Escribir en el fichero de disco la tabla de memoria que codifica la señal de 440 Hz. Se invocará a la función lejana *Write_WAV* del módulo.
- Cerrar el fichero de disco. Se invocará a la función lejana *fclose* del módulo.

El programa deberá hacer control de los posibles errores retornados por las funciones del módulo WAV. Para validar la corrección del programa, se reproducirá el archivo "*gen_la.wav*" mediante un reproductor multimedia de Windows y se comparará su sonido con el del archivo de referencia proporcionado con el material de esta práctica (*la.wav*).

Programa pract2b.asm (3 puntos)

Escribir un programa similar al del apartado anterior que permita generar un archivo WAV con un nombre y frecuencia de sonido que se solicitarán por teclado. Mediante sucesivas llamadas a este programa se generarán los siguientes archivos:

- "gen_do.wav" : nota Do (Middle C4) = 262 Hz
- "gen_re.wav" : nota Re (D4) = 294 Hz
- "gen_mi.wav" : nota Mi (E4) = 330 Hz
- "gen_fa.wav" nota Fa (F4) = 349 Hz
- "gen_sol.wav" : nota Sol (G4) = 392 Hz
- "gen_si.wav" : nota Si (B4) = 494 Hz
- "gen_do2.wav" : nota Do (Tenor C5) = 523 Hz

Cuando se introduzca la palabra “quit” como nombre, el programa deberá terminar.

El programa deberá hacer control de los posibles errores retornados por las funciones del módulo WAV. Para validar la corrección del programa, se reproducirán los archivos WAV generados y se comparará su sonido con el de los archivos correspondientes que se proporcionan con el material de esta práctica.

Se recomienda desarrollar una rutina de conversión *ASCII_to_DEC* para la conversión de datos numéricos introducidos desde el teclado.

Programa pract2c.asm (2 puntos)

Escribir un programa que lea la cabecera (*header*) de un fichero .WAV cuyo nombre se proporcione desde el teclado. El programa deberá mostrar en pantalla los campos más relevantes del mismo (ver anexo):

- ✓ Nombre del fichero
- ✓ Tipo de archivo (WAVE, etc.)
- ✓ Frecuencia de muestreo (*sample rate*)
- ✓ Número de muestras
- ✓ Número de canales
- ✓ Bytes por segundo (promedio)

En caso de leer un fichero erróneo, vacío o incompleto, se deberá imprimir el mensaje de error asociado a la anomalía detectada.

ENTREGA DE LA PRÁCTICA: Fecha y contenido

Se deberá subir a Moodle un fichero comprimido zip que contenga los ficheros fuente de los programas y el Makefile. El fichero comprimido sólo deberá ser subido por uno de los miembros de la pareja.

Los ficheros a entregar deberán contener en la cabecera los nombres de los autores y el identificador de la pareja. Así mismo, el código de los ficheros entregados deberá estar correctamente tabulado y comentado. La falta de comentarios o la baja calidad de éstos será calificada negativamente.

El límite de fecha de subida de los ficheros para cada grupo es el siguiente:

Grupos del Miércoles: 24 de Marzo de 2020 a las 23:55h
Grupos del Jueves: 25 de Marzo de 2020 a las 23:55h
Grupos del Viernes: 26 de Marzo de 2020 a las 23:55h

Anexo: Funciones públicas lejanas (FAR) del módulo externo WAV (*wav.obj*)

```
; ========================================================
;                                Init_WAV_Header
; Input:
;      DX = Sample rate
;      CX = Number of samples
; ========================================================
;                                fopen
; Input:
;      DS:DX = Address of ASCIIZ filename
; Output:
;      CF = Set on error
;      AX = File handle (CF=0) / Error code (CF=1)
; ========================================================
;                                fclose
; Input:
;      BX = File handle
; Output:
;      CF = Set on error
;      AX = Error code
; ========================================================
;                                fread
; Input:
;      BX = File handle
;      CX = Number of bytes to read
;      DS:DX = Address of buffer
; Output:
;      CF = Set on error
;      AX = Number of bytes read (CF=0) / Error code (CF=1)
; ========================================================
;                                Write_WAV
; Input:
;      BX = File handle
;      DS:DI = Address of WAV data chunk
; Output:
;      CF = Set on error
;      AX = Number of data bytes written (CF=0) / Error code (CF=1)
; ========================================================
;                                Read_WAV
; Input:
;      BX = File handle
;      DS:DI = Address of WAV data chunk
; Output:
;      CF = Set on error
;      AX = Number of data bytes read (CF=0) / Error code (CF=1)
```

Anexo: Formato WAV (RIFF)

Sección	Tamaño (bytes)	Descripción
RIFF	4	Contiene los códigos ASCII de las letras: 'R', 'I', 'F' y 'F'. RIFF significa formato de archivo para el intercambio de recursos (*Resource Interchange File Format*).
	4	Entero positivo de 32 bits que almacena el tamaño en bytes del resto del archivo. Es decir, tamaño total del archivo – 8 (4 bytes de RIFF, y 4 bytes de este número).
WAVE	4	Estos 4 bytes indican que el archivo almacena audio, contienen los códigos ASCII de las letras: 'W', 'A', 'V' y 'E'.
	4	Estos 4 bytes contienen los caracteres 'f', 'm', 't' y ' ', indican que hay una subsección de "WAVE" denominada "fmt ". Esta subsección almacena las características de la grabación.
	4	Entero positivo de 32 bits que indica el tamaño en bytes del resto del bloque. En caso de ser un número impar, será necesario considerar la existencia de un byte de relleno.
	2	Entero positivo de 16 bits que indica el tipo de grabación. Un 1 significa PCM.
	2	Número de canales. Cuando se usa un canal se habla de una grabación mono o monoaural; mientras que al uso de dos canales se le suele llamar estéreo.
	4	Frecuencia de muestreo expresada en Hz.
	4	Número promedio de bytes por segundo. Los programas para reproducir audio suelen estimar el tamaño de su ***buffer*** usando este dato.
	2	Alineamiento, corresponde con el número bytes usados en el archivo por cada muestra (si es el caso, se consideran ambos canales).
	2	Bits por muestra.
data	4	Estos 4 bytes contienen los códigos ASCII de las letras: 'd', 'a', 't' y 'a', indican que a continuación están los datos.
	4	Entero positivo de 32 bytes que indica el espacio en bytes que ocupan los datos.
	n	Datos.

Anexo: **Funciones del sistema operativo para imprimir texto y finalizar la ejecución.**

El sistema operativo MS-DOS facilita la mayor parte de sus servicios a través de la interrupción 21h. Antes de que nuestro programa invoque la llamada mediante la instrucción INT 21H, se debe cargar en el registro AH el número de la función solicitada. Además, cada función puede requerir de otros parámetros de entrada cuyos valores deberán almacenarse en otra serie de registros. A continuación, se detallan tres funciones de uso habitual en los programas que se desarrollan en el laboratorio.

Función 2H: Envío de un código ASCII al periférico pantalla

Descripción: Imprime un único carácter por pantalla.
Parámetros de entrada: AH = 2h.
DL = código ASCII del carácter que se imprimirá en pantalla.
Parámetros de salida: Ninguno.
Registros afectados: Ninguno**.**
Ejemplo:

```
mov ah, 2    ; Número de función = 2
mov dl, 'A'  ; Se desea imprimir la letra A
int 21h      ; Interrupción software del sistema operativo
```

Función 9H: Envío de una cadena de caracteres ASCII al periférico pantalla

Descripción: Impresión por pantalla de una cadena de caracteres que termina con el carácter '$'.
Parámetros de entrada: AH = 9h.
DX = Desplazamiento en memoria desde el origen del segmento de datos (DS) donde se ubica en memoria el primer carácter de la cadena.
Parámetros de salida: Ninguno.
Registros afectados: Ninguno**.**
Ejemplo:

```
.DATA
Texto DB "Hello world",13,10,'$' ; string terminado con los caracteres CR,LF y'$'
. CODE
.....
; Si DS es el segmento donde está el texto a imprimir:
mov dx, offset Texto    ; DX : offset al inicio del texto a imprimir
mov ah, 9               ; Número de función = 9 (imprimir string)
int 21h                 ; Ejecuta el servicio del sistema operativo
```

Función 0AH: Lectura de caracteres introducidos por el periférico teclado

Descripción: Captura los caracteres introducidos por el teclado, los imprime por pantalla a modo de eco local (local echo) y se los transfiere posteriormente a la aplicación. La función finaliza cuando el usuario introduce el ASCII 13 (Retorno de carro). El sistema operativo permitirá la entrada de caracteres siempre que no se supere el máximo permitido definido en uno de los parámetros de entrada. En el caso contrario, estos no se mostrarán por pantalla ni se almacenarán.

Parámetros de entrada: DX = Desplazamiento en memoria desde el origen del segmento de datos (DS) donde se ubica el espacio reservado para capturar los caracteres. En el primer byte de dicho espacio reservado se ha de almacenar el número de caracteres que se desea capturar.

Parámetros de salida: En el segundo byte de la zona de memoria apuntada por DX el sistema operativo almacena el número de caracteres almacenados. A partir del tercer byte, el sistema operativo almacena los caracteres tecleados.

Registros afectados: Ninguno**.**

Ejemplo:

```
MOV AH,0AH                  ;Función captura de teclado
MOV DX,OFFSET NOMBRE        ;Area de memoria reservada = etiqueta NOMBRE
MOV NOMBRE[0],60            ;Lectura de caracteres máxima=60
INT 21H
```

En NOMBRE[1] el sistema operativo almacenará el número de caracteres tecleados.

Función 4C00H: Fin de programa

Descripción: Esta función indica al sistema operativo que el proceso (programa) ha finalizado correctamente.

Parámetros de entrada: Ninguno.

Parámetros de salida: Ninguno.

Registros afectados: Ninguno**.**

Ejemplo:

```
mov ax, 4C00h  ; Fin de programa
int 21h        ; Ejecuta el servicio del sistema operativo
```