SISTEMAS OPERATIVOS

Lenguaje ensamblador en arquitecturas 8086/8088 parte 3

Elaborado por: Ukranio Coronilla

En muchas aplicaciones de lenguaje ensamblador se hace necesaria la lectura de datos provenientes del teclado y también la impresión de caracteres en pantalla. A diferencia del lenguaje C donde solo basta el uso de las funciones scanf y printf para lograrlo, en ensamblador es necesario mandar llamar a rutinas de interrupción. En este caso la interrupción puede transferir el control al código del sistema operativo DOS o al código almacenado en el chip de la tarjeta madre conocido como BIOS.
Como su nombre lo indica, una interrupción interrumpe de manera abrupta la ejecución del programa, para pasar a ejecutar algún procedimiento que se ha programado de manera explícita, para manejar dicha interrupción. En otras palabras el manejador de interrupción es un código asociado con un identificador, el cual se ejecuta cada que se manda a llamar una interrupción INT y se le pasa como parámetro dicho identificador.
El vector de interrupciones es una tabla de dos columnas que correlaciona el número de interrupción, con la dirección de memoria donde se encuentra ubicado el inicio del manejador de interrupción. Este vector se encuentra en las direcciones más bajas de memoria, y es accesado cada que se llama a una interrupción.

DESARROLLO EXPERIMENTAL

Ejecute el siguiente programa PROG3_1.ASM, el cual solicita al usuario que introduzca un nombre y después lo despliega al centro de la pantalla.

```
1 PILASG SEGMENT PARA STACK 'Pila'
2 DW 64 DUP(0)
3 PILASG ENDS

4 DATOSG SEGMENT PARA PUBLIC 'Datos'
5 NAMEPAR LABEL BYTE
6 MAXNLEN DB 20
7 NAMELEN DB ?
8 NAMEFLD DB 21 DUP(' ')
9 PROMPT DB 'Nombre? ', '$'
10 DATOSG ENDS

11 CODIGO SEGMENT PARA PUBLIC 'Codigo'
12 MAIN PROC FAR
13 ASSUME SS:PILASG, DS:DATOSG, CS:CODIGO
14 MOV AX, DATOSG
15 MOV DS, AX
16 MOV ES, AX
17 CALL Q10CLR
18 A20LOOP:
19 MOV DX, 0000
20 CALL Q20CURS
21 CALL B10PRMP
```

```
22 CALL D10INPT
23 CALL Q10CLR
24 CMP NAMELEN, 00
25 JE A30
26 CALL E10CODE
27 CALL F10CENT
28 JMP A20LOOP
29 A30:
30 MOV AX, 4C00H
31 INT 21H
32 MAIN ENDP

33 B10PRMP PROC NEAR
34 MOV AH, 09H
35 LEA DX, PROMPT
36 INT 21H
37 RET
38 B10PRMP ENDP

39 D10INPT PROC NEAR
40 MOV AH, 0AH
41 LEA DX, NAMEPAR
42 INT 21H
43 RET
44 D10INPT ENDP

45 E10CODE PROC NEAR
46 MOV BH, 00
47 MOV BL, NAMELEN
48 MOV NAMEFLD[BX], 07
49 MOV NAMEFLD[BX + 1], '$'
50 RET
51 E10CODE ENDP

52 F10CENT PROC NEAR
53 MOV DL, NAMELEN
54 SHR DL, 1
55 NEG DL
56 ADD DL, 40
57 MOV DH, 12
58 CALL Q20CURS
59 MOV AH, 09H
60 LEA DX, NAMEFLD
61 INT 21H
62 RET
63 F10CENT ENDP

64 Q10CLR PROC NEAR
65 MOV AX, 0600H
66 MOV BH, 30
67 MOV CX, 0000
68 MOV DX, 184FH
69 INT 10H
70 RET
71 Q10CLR ENDP

72 Q20CURS PROC NEAR
73 MOV AH, 02H
74 MOV BH, 00
75 INT 10H
76 RET
77 Q20CURS ENDP
```

```
78 CODIGO ENDS
79 END MAIN
```

Una interrupción es una llamada generada por el hardware o por el software (debida a una instrucción). Toda interrupción interrumpirá el programa, porque llamará a un procedimiento para servicio de interrupción o un manejador de interrupción.
En este programa se hace uso de interrupciones (INT) para el manejo de E/S. Específicamente la INT 10H del BIOS para manejar la pantalla y la INT 21H del DOS para mostrar salidas en pantalla y aceptar entradas desde el teclado. Las interrupciones de DOS pueden manejar tareas más complejas y transferir el control de manera automática al BIOS, quien maneja la parte de bajo nivel en la operación.
La INT 10H permite ubicar el cursor en cualquier posición y limpiar la pantalla, mediante los valores 02H y 06H respectivamente. En el caso de la INT 21H del DOS se tienen las capacidades mostradas en la tabla 3.1.

02H, 09H y 40H	Despliegue en pantalla
0AH y 3FH	Entrada desde el teclado

Tabla 3.1

Para escribir en la pantalla se tiene una malla de posiciones direccionables, con 25 renglones y 80 columnas como se muestra en la tabla 3.2.

	Renglón	Columna	Renglón	Columna
Esquina superior izquierda	00	00	00H	00H
Esquina superior derecha	00	79	00H	4FH
Centro	12	39/40	0CH	27H/28H
Esquina inferior izquierda	24	00	18H	00H
Esquina inferior izquierda	24	79	18H	4FH

Tabla 3.2

Existe un área de despliegue de video o buffer, que inicia en la localidad de BIOS B0000H y permite utilizar 4K de memoria: 2K para caracteres y 2K para atributos de carácter.

La línea 5 se utiliza en la función 0AH de la INT 21H(véase líneas 39 a 44) para aceptar datos desde el teclado. Para que la interrupción funcione, se requiere de una lista de parámetros que contenga los campos que va a procesar la interrupción. Por ejemplo, necesita conocer la longitud máxima de los datos que se van a ingresar, posteriormente la función envía el número de bytes que realmente se introdujeron. En la línea 5 la función carga la dirección donde inicia la lista de parámetros, es decir NAMEPAR. La línea 6 contiene el número máximo de caracteres que se van a ingresar. La línea 7 almacena el número real de caracteres introducidos y la línea 8 inicia un campo que contiene los caracteres tecleados, de izquierda a derecha.

Ejercicio 1

Según la línea 6 ¿Cuál es el máximo número de caracteres de entrada que se podrían pedir en esta línea? Con turbo debugger observe al ejecutar el programa que se almacena en las variables

indicadas en las líneas 5, 6, 7, 8 y 9. Determine cuáles son los caracteres ascii que se almacenan por omisión al final del arreglo.

La adquisición de los datos se hace dentro del procedimiento o función D10INPT ubicado en la línea 39. La línea 41 carga la dirección que contiene la lista de parámetros declarada en NAMEPAR.

Un procedimiento se define con la siguiente sintaxis:

nombre_procedimiento PROC [FAR, NEAR]

FAR se utiliza cuando el procedimiento se encuentra en cualquier localidad de memoria del sistema, y NEAR cuando el procedimiento se encuentra en el mismo segmento de código que el programa como en la línea 45. Todo procedimiento debe terminar con:

nombre_procedimiento ENDP

Como sucede en las líneas 20 a 23, un procedimiento se manda llamar mediante la instrucción:

CALL *nombre_procedimiento*

Dentro de un procedimiento, la instrucción RET regresa al lugar desde donde se le mandó llamar con la instrucción CALL.

Al ejecutar el programa, antes de solicitar el nombre aparece el texto “Nombre?”, y para desplegar estos caracteres se llama al procedimiento `B10PRPM` en la línea 33. Observe en este caso que se utiliza la interrupción 21 con el valor 09H para despliegue de pantalla. La variable `PROMPT` (línea 9) incluye también el signo de pesos, lo cual indica que finaliza el despliegue. Se puede también poner el signo $ dentro de las comillas así:

```
'Nombre? $'
```

La instrucción `LEA` (línea 35) carga la dirección de la cadena de despliegue en DX. Si al final de la cadena se omite el signo de pesos, la operación despliega caracteres de la memoria, hasta que se encuentre un signo de $ (si existe alguno).

El primer procedimiento llamado por la función principal `Q10CLR`, limpia la pantalla. Para limpiar la pantalla se utiliza la función 06H de la INT 10H del BIOS (las interrupciones desde la 20H hasta la 3FH están reservadas para operaciones del DOS). Se puede limpiar un área específica de la pantalla dando el renglón y columna iniciales, hasta el renglón y columna finales como se indica (véase la tabla 3.2):

CX = Renglón (1 byte) Columna (1 byte) iniciales
DX = Renglón (1 byte) Columna (1 byte) finales

BH contiene el número del atributo, por ejemplo 71H indica color blanco (7) sobre azul (1) como se muestra en la tabla 3.3.

Color	ID	Color	ID
Negro	0	Gris	8
Azul	1	Azul claro	9
Verde	2	Verde claro	A
Cian	3	Cian claro	B
Rojo	4	Rojo claro	C
Magenta	5	Magenta claro	D
Café	6	Amarillo	E
Blanco	7	Blanco brillante	F

Tabla 3.3

El segundo procedimiento llamado `Q20CURS` coloca el cursor desde donde será desplegado el siguiente carácter. Para lo cual se utiliza la función 02H de la INT 10H.

Se coloca en BH el número de página o pantalla (por lo común 0) y en DX el renglón y columna requeridos. El tercer procedimiento se explicó con anterioridad, a continuación el procedimiento `D10INPT` en la línea 39 permite obtener el nombre proveniente del teclado. La función 0AH de la INT 21H se utiliza para este objetivo.

La instrucción `CMP` en la línea 24 compara 2 campos de datos, uno o ambos están contenidos en un registro y tiene la siguiente sintaxis:

CMP *destino, fuente*

Si el operando destino es igual al fuente, ZF = 1(la bandera del cero está activa) y CF = 0. Si el operando fuente es mayor que el destino, entonces ZF = 0 y CF = 1(la bandera de acarreo se activa). Por último, si el operando fuente es menor que el destino, ZF = 0 y CF = 0.

Inmediatamente después de comparar se utilizan instrucciones de salto como JE, JZ. En este caso se utiliza JE en la línea 25 y salta a la etiqueta A30 de la línea 29. El salto se realiza si la última operación provoca que la bandera ZF = 1.

La instrucción JMP en la línea 28 hace un salto incondicional hacia el lugar donde se encuentra la etiqueta `A20LOOP`.

En el caso que el tamaño de la cadena sea diferente de cero, continua en la línea 26, donde se manda llamar al procedimiento `E10CODE`. Este procedimiento almacena en BX el número de caracteres que introdujo el usuario a la cadena `NAMEFLD` y reemplaza el caracter ENTER (0D) con el byte que corresponde en ASCII a la campana, además de añadir el caracter delimitador $.

Posteriormente se manda llamar al procedimiento F10CENT en la línea 52. La instrucción `SHR` en la línea 54 hace un corrimiento de n bits hacia la derecha, por ejemplo: Si `AL = 01101011` y se ejecuta `SHR AL, 2`. Entonces se recorren todos los bits hacia la derecha 2 posiciones quedando `AL = 00011010`. Esta operación corresponde a dividir el numero entre 2 e ignorar las cifras decimales.

La instrucción `NEG` niega todos los bits almacenados en el byte DL. El negar los bits y sumarles 40 equivale a restarle el número negado a 40.

Observe que básicamente las líneas 53 a 57 inicializan la posición, para que el texto aparezca en el centro de la pantalla. En la línea 57 se fija el cursor, y con las líneas posteriores se despliega la cadena de caracteres almacenados.