Universidad Tecnológica de la Mixteca
Lenguaje Ensamblador
Frida Ximena Martínez Lorenzo
Profesora:M.C. Everth Haydee Rocha Trejo
Tercera Evaluación Parcial
Ingeniería en Computación || Grupo: 402-B

09 de junio de 2023

A lo largo de este documento se propone documentar el funcionamiento del programa de mouse.asm en lenguaje ensamblador 8086 brindado por la profesora que imparte la materia. :

Para comenzar, adoptamos la macro que tiene el protocolo de la directiva corta. En esta parte nos encargamos de inicializar el segmento de datos, ya que tanto este como el ES son responsabilidad del programador, puesto que  el cargador de DOSBOX se encarga de inicializar el Segmento de Pila y el Segmento de Código.

```
INICIO_DC MACRO

        PUSH DS
        SUB AX,AX
        PUSH AX
        MOV AX,@data
        MOV DS,AX

ENDM
```

Para este programa es necesario alinear el segmento DS, con el ES. Esto no se consideró en la macro del protocolo de directiva corta ya que no es utilizada en todos los programas. Por lo tanto se añade esta línea:

```
MOV ES,AX
```

Lo siguiente es inicializar el modo texto, para esta acción nos ayudamos de la MACRO de INITTEXT, la cual utilizando el servicio 00h de la INT 10 nos permite hacer una inicialización del modo texto al colocar en ah el modo 03.

```
INITTEXT MACRO
        PUSH AX
        MOV AH,00H
        MOV AL,03H
        INT 10H
        POP AX
ENDM
```

Después de esto vemos involucradas las cuatro líneas de código del lado derecho, las cuales involucran a tres variables inicializadas en el segmento de datos como lo vemos en la parte izquierda, donde YXI y YXF son del tamaño de una doble palabra, y FPP es de tamaño de un byte.

```
19 YXI DW 00
20 YXF DW 00
21 FPP DB 00
22
```

```
30
31 MOV FPP,3EH
32 MOV YXI,00H
33 MOV YXF,184FH
34 TEXTBACKGROUND FPP,YXI,YXF
35
```

Las variables están fungiendo papeles específicos, primeramente FPP está siendo el encargado de llevar la información sobre el fondo y el color del frente (letras por decir un ejemplo), YXI por otro lado es el encargado de tener el renglón y columna de inicio, es decir de donde empezará a colorear la pantalla en este caso desde el 00; finalmente YXF se encarga de almacenar el renglón y columna final (hasta donde hemos de limpiar la pantalla). Lo explico de esta forma específica puesto que en la MACRO DE TEXTBACKGROUND la cual se anexará en breve ocupa el servicio 06 de la interrupción 10H la cual se encarga de recorrer la pantalla hacia arriba.

Se considera un limpiado puesto que las partes seleccionadas de las pantallas serán sustituidas de lo que había por lo que se coloque en BH que sería la nueva base que se ha de colocar, y sobre la cual se habrá que trabajar.

```
TEXTBACKGROUND MACRO FL,INI,FIN
   PUSH AX
   PUSH BX
   PUSH CX
   PUSH DX
   MOV AX,0600H     ;SERVICIO EN AH, AL LINEAS A RECORRER 00 PANT COMPLETA
   MOV BH,FL        ;FONDO Y LETRAS
   MOV CX,INI       ;REN:COL INI
   MOV DX,FIN       ;REN:COL FIN
   INT 10H
   POP DX
   POP CX
   POP BX
   POP AX
ENDM
```

```
36 I_MOUSE
37 CMP AX,00
38 JE ERROR
39 M_MOUSE
40 MOV YXI,1843H
```

En las líneas siguientes pasamos primeramente a realizar la inicialización del Mouse mediante una Macro, y es que se sabe que es necesario hacer esta acción cuando se planea realizar la interacción mediante el mouse.

```
I_MOUSE MACRO
      MOV AX,00
      INT 33H
ENDM
```

Mediante el servicio 00H de la INT10 se permite que el cursor se posicione en el centro de la pantalla, se oculte el mismo, se calibra el puntero en el modo que se está utilizando, se establece el umbral de velocidad doble a 64 mickey por seg, se establece la razón mickey a pixel.

Se compara el registro AX con 00, ya que de ser cero significa que el servicio ha retornado que el controlador no se encuentra disponible, por lo cual habría de marcar un error.

El MACRO de M_MOUSE nos permite visualizar el cursor.

```
M_MOUSE MACRO
        PUSH AX
        MOV AX,01H
        INT 33H
        POP AX
ENDM
```

Finalmente situamos YXI en el renglón y columna donde se habrá de situar el marcador mediante el cual se mostrarán las coordenadas del cursor no en pixeles, sino traducidas.

```
OTRO:
    CALL P_MOUSE ; SERVICIO 03
    CMP BX,01
    JE EXIT
```

Posteriormente, se manda a llamar al procedimiento P_MOUSE, en este se hace el llamado del servicio 03H de la interrupción 33H la cual nos retorna información sobre el mouse, como lo es la pulsación de los botones, las coordenadas del cursor.

```
72 P_MOUSE PROC NEAR
73 SAL3: MOV AX,03
74 INT 33H
75 CMP BX,01
76 JE SAL1
```

```
90 SAL1:
91     RET
92 P_MOUSE ENDP
```

Se compara el registro BX ya que es en este donde se almacena la información sobre las pulsaciones, el bit menos significativo nos habla sobre el botón izquierdo del mouse, en la comparación se busca saber si esta arriba o abajo, donde abajo es primido, de ser así salta a la etiqueta sal1.

```
78      MOV AX,CX
79      MOV CL,3
80      SHR AX,CL
81      SHR DX,CL
82      MOV CX,AX
83      CMP CX,XB
84      JNE SAL2
85      CMP DX,YB
86      JE SAL3
87  SAL2:
88      MOV XB,CX
89      MOV YB,DX
```

Luego de ello, se trabaja con las coordenadas, la horizontal que se encuentra en CX y la vertical que se encuentra en DX, se respalda el dato de las coordenadas horizontales en AX, coloca el CL el valor de 03 ya que el corrimiento a la derecha compromete el registro. Se hace el corrimiento a la derecha de tres bits, esto con el propósito de transformar los píxeles originales dados en una escala que sea más manejable,como lo es el decimal y es que es el formato en el cual el usuario conoce las dimensiones de la pantalla 80x25.

En caso de que no sean iguales a las coordenadas anteriores se actualizarán a las nuevas, las coordenadas son almacenadas en XB y YB (Columna y Renglón)

```
43      CALL P_MOUSE ; SER
44      CMP BX,01
45      JE EXIT
46      GOTOXY YXI
```

La comparación de BX, 01 es debido a que de haberse detectado en el procedimiento se salía del mismo para regresar al programa principal y es que en caso de que se oprimiera el botón izquierdo del mouse se debía de salir del programa.

Pasando a GOTOXY, tenemos que:

```
GOTOXY MACRO REN_COL
      PUSH AX
      PUSH BX
      PUSH DX
      MOV AH,02H  ;POSICIONAR CURSOR
      MOV BH,00H  ;PAGINA O PANTALLA
      MOV DX,REN_COL  ;REN Y COL
      INT 10H
      POP DX
      POP BX
      POP AX
ENDM
```

Esta macro recibe como parámetro el renglón y la columna donde se ha de posicionar el cursor, en este caso recibe YXI que contiene las coordenadas de donde se localiza el marcador de columna y renglón.

```
46      MOV AX,XB
47      CALL CONV
```

En esta parte del código se hace el tratamiento de los datos antes de imprimir en pantalla. Como paso inicial cargamos a AX el contenido de XB que es las coordenadas de X en decimal, esto debido a que el procedimiento de CONV trabaja con el dato recibido en AX.

```
94  CONV PROC NEAR
95      MOV VALASC,2020H
96      MOV CX,10
97      LEA SI,VALASC+1
98      CMP AX,CX
99      JB C1
100     DIV CL
101     OR AH,30H
102     MOV [SI],AH
103     DEC SI
104     C1: OR AL,30H
105     MOV [SI],AL
106     RET
107 CONV ENDP
```

CONV se encarga de transformar el contenido recibido en AX en su representación es ASCII, el cual almacena el dato transformado en VALASC, el cual se inicializa con espacios en blanco, se posiciona a SI en el segundo byte de VALASC, se asume ahí ha de ir el bit menos significativo, nos localizamos ahí ya que de ser un número entre el 0-9 debería de hacerse un pase directo y no traducir el 0 por lo cual de esta forma el número en cuestión se colocarían en el lugar apropiado.

Por otro lado en caso de ser un número de dos cifras, se ha dividido entre la base del número es decir 10, con esto lograremos separar cada cifra, tener el bit más significativo en AL (el cociente) y el bit menos significativo en AH(el residuo). Primero se guardaría el menos significativo, se posiciona SI en la parte del más significativo y se guarda el más significativo, quedando en su ASCII.

```
48      MOV AX,VALASC
49      MOV XASCII,AX
50      MOV AX,YB
51      CALL CONV
52      MOV AX,VALASC
53      MOV YASCII,AX
```

Luego, respaldamos el dato de VALASC en XASCII mediante AX, ya que este es cambiante por cada vez que se invoque al procedimiento de CONV. Una vez asegurado el primer dato se procede al tratamiento de la variable con las coordenadas del eje de las Y, este dato será convertido a ASCII y luego respaldado en YASCII.

```
54      DESPLIEGA DESPDATO ;SERVICIO 40 DE LA INT 21
55      JMP OTRO
```

```
DESPLIEGA MACRO DATO
        MOV AH,40H
        MOV BX,01
        MOV CX,11
        LEA DX,DATO
        INT 21H
ENDM
```

Haciendo un paréntesis hablemos del **servicio 40H de la INT 21H**, la cual nos permite mostrar texto en la pantalla, tiene comprometidos los registros de la siguiente manera:

| REGISTRO | CONTENIDO |
|---|---|
| **AH** | 40H |
| **BX** | Número de controlador de archivo. Se establece el 01H para la salida estándar. |
| **CX** | Longitud del texto a mostrar en bytes. |
| **DX** | Apunta a la parte de la memoria donde se encuentra el texto que se quiere mostrar en pantalla |

RETORNO:

| | REGISTROS |
|---|---|
| **CF = 0** | PROCEDIMIENTO EXITOSO<br><br>AX = Número de bytes transferidos<br>AX = 0 significa que el disco estaba lleno |
| **CF = 1** | PROCEDIMIENTO FALLIDO<br>AX = ERROR<br><br>5: ACCESO DENEGADO<br>6: Controlador inválido ó no se pudo abrir |

La macro de DESPLIEGA recibe un dato, el cual es el que se ha de mostrar en pantalla, que en este caso es DESPDATO, el cual se encarga de unir los símbolos como “X =” y “Y =” a sus valores correspondientes convertidos a ASCII en breves líneas anteriores.

El JMP OTRO, nos indica que se seguirá mostrando los datos correspondientes al seguimiento del cursor hasta que el usuario decida darle click izquierdo, que es cuando saltará a la parte de fin de programa.

```
61
62 EXIT:
63     F_MOUSE
64     MOV FPP,07h
65     MOV YXI,00H
66     MOV YXF,184FH
67     TEXTBACKGROUND FPP,YXI,YXF
68     ;SAL_A_DOS
69     EXIT_PROGRAMA
70     RET
71 MAIN ENDP
```

Insertamos el código correspondiente a la MACRO de F_MOUSE:

```
F_MOUSE MACRO
        PUSH AX
        MOV AX,02H
        INT 33H
        POP AX
ENDM
```

Mandando a llamar al servicio 02H de la INT 33H lo que nos permite hacer es esconder el puntero del mouse.

Posteriormente usamos la macro de TEXTBACKGROUND para regresarlo a los colores originales del modo texto el cual es negro y blanco, el YXF nos indica que es toda la pantalla

Y para culminar usamos la macro de que tiene el servicio  4CH de la INT 21 la cual nos permite salir de DOSBOX y terminar la emulación.

**NOTA:** Los procedimientos de C_MOUSE Y C_CURSOR fueron eliminadas ya que no se utilizaban en el código.