UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERIAS

Actividad 11

Alumno

Alejandro Covarrubias Sánchez

Código

221350192

Profesor

José Juan Meza Espinoza

Carrera

Ingeniería en Computación

Materia

Seminario de Solución de Problemas de Traductores de Lenguajes I

Fecha de entrega

20 de noviembre de 2023

LECTURA DE MOUSE Y ALGORITMO DE LINEA RECTA DE BRESENHAM

Inicio del programa: declaración de variables y macros

```
org 100h
jmp inicio

     x1 db 0
     y1 db 0
     x2 db 0
     y2 db 0
     Ddx db ?
     Ddy db ?
     2dy db ?
     Dold db ?
     x db ?
     y db ?
     oldX db -1
     oldY db 0
     ddyx db 0
     msg db " Presiona cualquier tecla....      $"

PIXEL macro x,y,color
     pusha
     mov cl,x          ;Coordenada en X
     mov dl,y          ;Coordenada en Y
     mov al,color      ;Color
     mov ah,0Ch        ;Escribe un punto en pantalla

     int 10h
     popa
endm
```

Primer segmento del código: Inicio del programa y obtención de coordenadas

```
inicio:
     mov ah,0          ;Modo de video
     mov al,13h        ;Pantalla de 320 x 200
     int 10h

     mov ax,0
     int 33h
     cmp ax,0

check_mouse_button:
     mov ax, 3         ;Leer el estado de los botones y la posicion
                       ;del cursor (Devuelve la posicion del cursor
                       ;y el estado de sus botones)
                       ;cx = posicion columna (H) y dx = posicion fila (V)
     int 33h
     shr cx,1          ;x/2 - en este modo se reduce el valor de CX a la mitad.
     cmp bx,1
     jne xor_cursor:
     mov al,1010b      ;color de pixel (Verde Claro)
     cmp x,0
     je draw_initial_coord
     cmp x2,0
     je draw_end_coord

draw_initial_coord:
     mov x,cl
     mov y,dl
     jmp draw_pixel

draw_end_coord:
     mov x2,cl
     mov y2,dl
     jmp draw_pixel
```

Segundo segmento del código: Bucle de ejecución

```
xor_cursor:
     cmp oldX, -1
     je not_required
     push cx
     push dx
     mov cl, oldX
     mov dl, oldY
     mov ah, 0dh          ;Lectura de un punto
                          ;DX = numero de region
                          ;CX = numero de columna
                          ;AL = punto leido
     int 10h
     xor al, 1111b        ;Color del pixel
     mov ah, 0ch          ;Dibuja pixel
     int 10h
     pop dx
     pop cx


not_required:
     mov ah, 0dh          ;Lectura de un punto
     cmp x2,0
     ja Bresenham         ;DX = numero de region
                          ;CX = numero de columna
                          ;AL = punto leido
     int 10h
     xor al, 1111b        ;Color del pixel Blanco Brillante
     mov oldX, cl
     mov oldY, dl


draw_pixel:
     mov ah, 0ch          ;Dibuja pixel
     int 10h              ;DX = numero de renglon  (fila)
                          ;CX = numero de columna


check_esc_key:
     mov dl, 255
     mov ah, 6            ;I/O directo de consola
     int 21h              ;Espera hasta recibir un caracter
                          ;proveniente del teclado
     cmp al, 27           ;esc?
     jne check_mouse_button
```

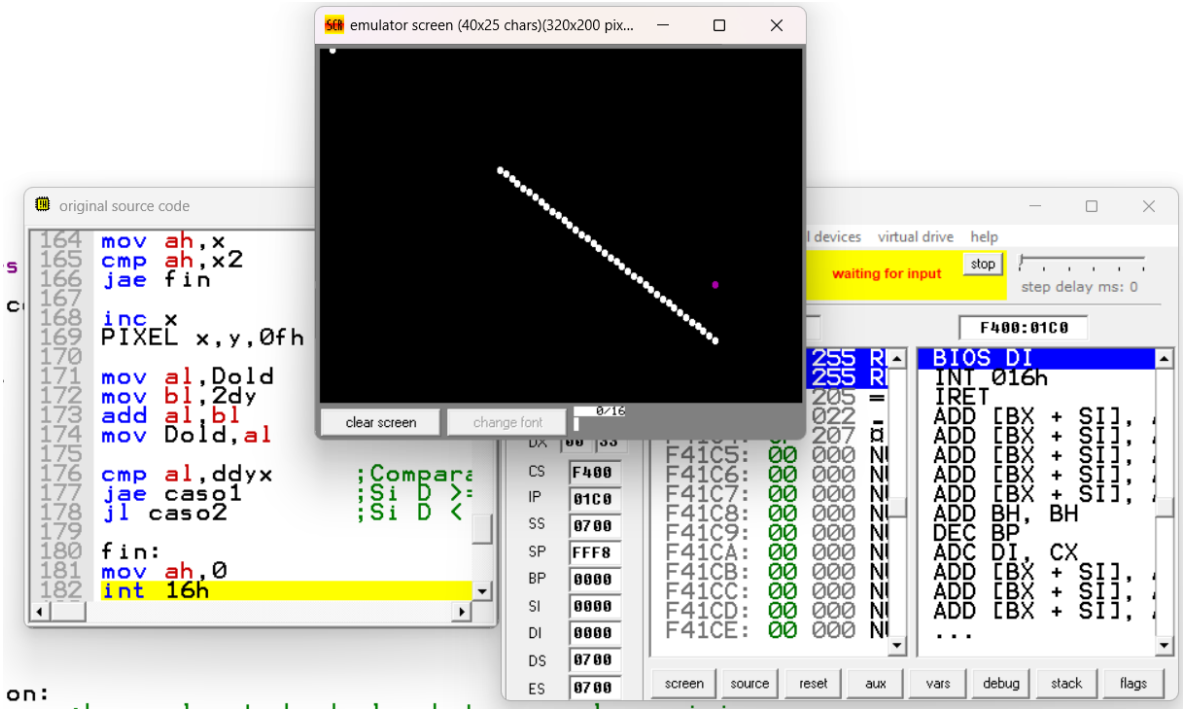
Tercer segmento del código: Algoritmo de Bresenham y fin del programa

```
Bresenham:
     mov al,x
     mov bl,y
     mov x1,al
     mov y1,bl

     PIXEL x,y,0fh

     mov al,x2
     mov bl,x1
     sub al,bl
     mov Ddx,al          ;Calcula la diferencia en x
                         ;entre los puntos inicial y final

     mov al,y2
     mov bl,y1
     sub al,bl
     mov Ddy,al          ;Calcula la diferencia en y
                         ;entre los puntos inicial y final

     mov al,Ddy
     mov bl,2
     mul bl
     mov 2dy,al
     sub al,Ddx
     mov Dold,al         ;Calcula el valor inicial de D

     cmp al,ddyx         ;Compara D con ddyx
     jae caso1           ;Si D >= ddyx, salta a caso1
     jl caso2            ;Si D < ddyx, salta a caso2
```

```
caso1:
     mov ah,x
     cmp ah,x2
     jae fin             ;Si hemos llegado al final en x, salta a fin

     inc x
     inc y
     PIXEL x,y,0fh       ;Dibuja el pixel siguiente

     mov al,Ddy
     mov bl,Ddx
     sub al,bl
     mov bl,2
     mul bl
     mov bl,Dold
     add al,bl
     mov Dold,al         ;Calcula el nuevo valor de D

     cmp al,ddyx         ;Compara D con ddyx
     jae caso1           ;Si D >= ddyx, salta a caso1
     jl caso2            ;Si D < ddyx    , salta a caso2
```

```
caso2:
     mov ah,x
     cmp ah,x2
     jae fin              ;Si hemos llegado al final en x, salta a fin

     inc x
     PIXEL x,y,0fh        ;Dibuja el pixel siguiente

     mov al,Dold
     mov bl,2dy
     add al,bl
     mov Dold,al

     cmp al,ddyx          ;Compara D con ddyx
     jae caso1            ;Si D >= ddyx, salta a caso1
     jl caso2             ;Si D < ddyx, salta a caso2

fin:
     mov ah,0
     int 16h
     ret
```

EJECUCIÓN



CONCLUSIONES:

En esta actividad volvimos a trabajar con el algoritmo de Bresenham, diseñado para dibujar una línea recta en pantalla, con el cual trabajamos en la actividad 9, por lo que aún lo teníamos fresco en la memoria y fue más sencillo de implementar. Lo que se me complicó fue poder implementar la lectura de la posición del mouse, pues tuve varios errores con Emu8086. El programa de ejemplo provisto por el profesor realmente me ayudó a comprender el método.

BIBLIOGRAFÍA

Brey, B. B. (2006). *Microprocesadores Intel* (Séptima ed.). Pearson.