MANUAL TECNICO

ARQUITECTURA DE COMPUTADORAS 1

Cristian Daniel Gomez Escobar
202107190

```
;; MODO VIDEO ;;
mov AH, 00
mov AL, 13
int 10
jmp pantalla_ini
```

Con esto activaremos el modo video del proyecto, esto permitirá dar la interfaz grafica de el programa

MENU PRINCIPAL

```
pepe:
        ;;;;;;;;;;;;;;;;
        call menu_principal
        mov AL, [opcion]
        ;; > INICIAR JUEGO
        cmp AL, 0
        je cargar_nivel_00

        ;; > CARGAR NIVEL
        cmp AL, 1
        je pedir_de_nuevo_nivel
        ;; > CONFIGURACION
        cmp AL, 3
        je config
        ;; > PUNTAJES ALTOS
        ;; > SALIR
        cmp AL, 4
        je fin
```

Con esto podremos elegir la opción que necesitamos mostrar, al principio se llamara a la instancia menú_principal

```
menu_principal:
    ;   mov DI, mapa
    ;   mov CX, 0005
    ;   call memset
        call clear_pantalla
        mov AL, 0
        mov [opcion], AL      ;; reinicio de la variable de salida
        mov AL, 5
        mov [maximo], AL
        mov AX, 50
        mov BX, 28
        mov [xFlecha], AX
        mov [yFlecha], BX
        ;; IMPRIMIR OPCIONES ;;
        ;;;; INICIAR JUEGO
        mov DL, 0c
        mov DH, 05
        mov BH, 00
        mov AH, 02
        int 10
        ;; <<-- posicionar el cursor
        push DX
        mov DX, offset iniciar_juego
        mov AH, 09
        int 21
        pop DX
```

En esta instancia se utilizara para mostrar las opciones disponibles que estén, esto será indicado por una flecha hecha con una matriz.

DISEÑOS

```
dim_sprite_caja_objetivo   db   08, 08
data_sprite_caja_objetivo  db  31,21,21,21,21,21,21,31
                           db  21,1F,1F,21,21,21,21,21
                           db  21,1F,21,21,21,21,21,21
                           db  21,21,21,10,10,10,21,21
                           db  10,10,10,10,1E,10,10,10
                           db  1E,1E,1E,10,10,10,1E,1E
                           db  1E,1E,1E,1E,1E,1E,1E,1E
                           db  31,1E,1E,1E,1E,1E,1E,31


dim_sprite_jug    db   08, 08

data_sprite_jug   db   31, 10, 10, 31, 31, 31, 31, 10
                  db   31, 31, 2c, 2b, 31, 31, 31, 2b
                  db   31, 31, 31, 2c, 2c, 2c, 2c, 2b
                  db   2b, 2b, 31, 2c, 10, 2c, 2c, 10
                  db   2b, 2b, 31, 28, 2c, 2c, 2c, 2b
                  db   31, 2b, 31, 2c, 2b, 2b, 2b, 31
                  db   31, 2b, 2c, 2b, 2c, 2b, 2c, 31
                  db   31, 31, 2c, 2b, 2b, 2b, 2b, 31

dim_sprite_jug_eq   db   08, 08
data_sprite_jug_eq   db   31, 10, 10, 31, 31, 31, 31, 10
                     db   31, 28, 2c, 2b, 31, 31, 28, 2b
                     db   31, 31, 28, 2c, 2c, 2c, 2c, 2b
                     db   2b, 2b, 31, 2c, 10, 2c, 2c, 10
                     db   2b, 2b, 31, 28, 2c, 2c, 2c, 2b
                     db   31, 2b, 31, 2c, 2b, 2b, 2b, 31
                     db   31, 2b, 2c, 2b, 2c, 2b, 2c, 31
                     db   31, 31, 2c, 2b, 2b, 2b, 2b, 31
```

Aquí hay un ejemplo del como se muestran los sprit, esto se utilizaran para generar las gráficas que desea el desarrollador

JUEGO

```
ciclo_juego:
    ;   mov AX,[puntos_juego]
    ;   call numAcadena

        mov DL, 3
        mov DH, 16
        mov BH, 00
        mov AH, 02
;   ;   mov bl, 0eh
        int 10
;       ;; <<-- posicionar el cursor
;       push DX
;       mov DX, offset numero2
;       mov AH, 09
;       int 21
;       pop DX

    mov al, puntos_juego
    add al, '0'  ; Convertir a carácter ASCII

    ; Imprimir la variable en pantalla
    mov ah, 0Eh  ; Función de impresión en pantalla
    mov bh, 0    ; Página 0
    mov bl, 1Fh  ; Color de texto blanco sobre fondo azul
    int 10h



        call pintar_mapa
        call entrada_juego
        jmp ciclo_juego
```

En esta opción mostrares principalmente el puntaje del juego, el cual serán los pasos dados en el transcurso de la ejecución, posteriormente se pintara el mapa en el cual se interactuara con el usuario, se llamara la entrada_juego el cual posee todas las instrucciones que se pueden seguir durante la ejecución del juego

```
entrada_juego:
        mov AH, 01
        int 16
        jz fin_entrada_juego  ;; nada en el buffer de entrada
        mov AH, 00
        int 16
        ;; AH <- scan code
        cmp AH, [control_arriba]
        je mover_jugador_arr
        cmp AH, [control_abajo]
        je mover_jugador_aba
        cmp AH, [control_izquierda]
        je mover_jugador_izq
        cmp AH, [control_derecha]
        je mover_jugador_der
        cmp AH,[F2]
        je pausa
        cmp AH, 3c
        ret
```

La instancia entrada juego contrara de el llamado a las teclas las cuales ejecutaran las instrucciones del sprit del jugador

```
mover_jugador_arr:
        inc puntos_juego

        mov AH, [xJugador]; ah <- posicion del jugador
        mov AL, [yJugador]; AL <- posicion del jugador


        push AX ; guardar posicion de arriba
        call obtener_de_mapa; DL <- elemento en mapa
        pop AX; restaurar posicion de arriba
        cmp DL, EN_OBJETIVO
        je en_objetivo_arriba

        dec AL ; posicion de arriba
        push AX ; guardar posicion de arriba
        call obtener_de_mapa; DL <- elemento en mapa
        pop AX; restaurar posicion de arriba
        ;; DL <- elemento en mapa
        cmp DL, PARED; aqui hay pared
        je hay_pared_arriba ; saltar a hay_pared_arriba

        cmp DL, CAJA
        je hay_caja_arriba

        cmp DL, OBJETIVO
        je hay_objetivo_arriba

        cmp DL, CAJA_OBJETIVO
        je quitar_pokemaster_arriba

        mov [yJugador], AL ; no hay pared, mover jugador
```

Aquí se ejecutara la instrucción de la tecla en dirección hacia arriba, esta traspasara las posiciones de los jugadores para que se puedan manejar en la pantalla, en eta caso antes de posicionarlo en el lugar correcto verificara que no haya una pared, caja o objetivo delante de el, si se da el caso de que lo haya, se dirigirá a las instancias requeridas en el Código, este mismo método se repetirá para todos los demás movimientos que se realicen.

INGRESAR ARCHIVO

```
pedir_de_nuevo_nivel:
        call clear_pantalla

        mov DL, 0c; aqui se pone el color de la letra
        mov DH, 01; aqui se pone la posicion en y
        mov BH, 00; aqui se pone el color de fondo
        mov AH, 02; aqui se pone la funcion de la interrupcion
        int 10; aqui se llama a la interrupcion
        ;; <<-- posicionar el cursor
        push DX; aqui se guarda la posicion del cursor
        mov DX, offset ingresa; aqui se pone el mensaje
        mov AH, 09; aqui se pone la funcion de la interrupcion
        int 21; aqui se llama a la interrupcion
        pop DX; aqui se recupera la posicion del cursor

        mov DI, offset nivel_nombre; aqui se pone la direccion de la cadena
        mov CX, 0005; aqui se pone el tamaño de la cadena
        call memset; aqui se llama a la funcion para limpiar la cadena

        mov DX, offset buffer_nivel; aqui se pone la direccion de la cadena
        mov AH, 0a; aqui se pone la funcion de la interrupcion
        int 21; aqui se llama a la interrupcion
        ;;; verificar que el tamaño del codigo no sea mayor a 5
        mov DI, offset buffer_nivel; aqui se pone la direccion de la cadena
        inc DI; aqui se incrementa la direccion
        mov AL, [DI]; aqui se obtiene el tamaño de la cadena
        cmp AL, 00; aqui se compara el tamaño de la cadena
        je  pedir_de_nuevo_nivel; si es igual a 0, pedir de nuevo
        cmp AL, 32 ; si es mayor a 20, pedir de nuevo
        jb  aceptar_tamanio
        mov DX, offset nueva_lin
        mov AH, 09
        int 21
```

Aquí posicionaremos en pantalla X y Y para poder solicitar el nombre, luego limpiaremos el buffer para evitar que se guarde el dato y pasaremos a los siguientes siclos de aceptación

```
aceptar_tamanio:
        mov SI, offset nivel_nombre ; aqui se pone la direccion de la cadena
        mov DI, offset buffer_nivel ; aqui se pone la direccion de la cadena
        inc DI; aqui se incrementa la direccion
        mov CH, 00; aqui se pone el contador
        mov CL, [DI]; aqui se obtiene el tamaño de la cadena
        inc DI; aqui se incrementa la direccion
duplicar_nombre:; aqui se duplica el nombre
        mov AL, [DI]; aqui se obtiene el caracter
        mov [SI], AL; aqui se guarda el caracter
        inc SI; aqui se incrementa la direccion
        inc DI; aqui se incrementa la direccion
        loop duplicar_nombre; aqui se repite el ciclo
        mov DX, offset nueva_lin; aqui se pone el mensaje
        mov AH, 09; aqui se pone la funcion de la interrupcion
        int 21; aqui se llama a la interrupcion
buscar_archivo:; aqui se busca el archivo
        mov CX, 26; aqui se pone el tamaño de la cadena
        mov DX, offset nivel_nombre; aqui se pone la direccion de la cadena
        mov AH, 40; aqui se pone la funcion de la interrupcion
        int 21; aqui se llama a la interrupcion
cargar_un_nivel_x:; aqui se carga un nivel
        mov AL, 00; aqui se pone el numero del nivel
        mov DX, offset nivel_nombre; aqui se pone la direccion de la cadena
        mov AH, 3d; aqui se pone la funcion de la interrupcion
        int 21; aqui se llama a la interrupcion
        jc inicio
        mov [handle_nivel], AX
        jmp ciclo_lineas
```