MANUAL TÉCNICO
PROYECTO 2
RACER CARS GAME

Ciudad de Guatemala 09 de Julio del 2019

ENSAMBLADOR UTILIZADO

MASM

El Microsoft Macro Assembler (MASM) es un ensamblador para la familia x86 de microprocesadores. Fue producido originalmente por Microsoft para el trabajo de desarrollo en su sistema operativo MS-DOS, y fue durante cierto tiempo el ensamblador más popular disponible para ese sistema operativo. El MASM soportó una amplia variedad de facilidades para macros y programación estructurada, incluyendo construcciones de alto nivel para bucles, llamadas a procedimientos y alternación (por lo tanto, MASM es un ejemplo de un ensamblador de alto nivel). Versiones posteriores agregaron la capacidad de producir programas para los sistemas operativos Windows. MASM es una de las pocas herramientas de desarrollo de Microsoft para las cuales no había versiones separadas de 16 bits y 32 bits.

La competencia

Hay en curso muchos desarrollos de proyectos de software que soportan el MASM, incluyendo IDEs (como RadASM y WinAsm Studio), depuradores (como OllyDbg), y desensambladores (incluyendo IDAPro, el desensamblador interactivo). El proyecto MASM32 ha puesto juntos una muy impresionante librería de programador, un repositorio de ejemplos de código, y una extraordinaria documentación para los usuarios del MASM. MASM también es soportado por una gran cantidad de páginas web y foros de discusión, incluyendo A pesar de la edad de este producto, sigue siendo uno de los ensambladores en existencia mejor soportados.

Soporte

Aunque existen rumores de que Microsoft ya no soporta al MASM, la verdad es que Microsoft *continúa* soportando este producto, incluso aunque fuera solamente para sus propios propósitos. Generalmente, el soporte es limitado solamente a agregar nuevas instrucciones a medida que aparecen nuevos procesadores y a mejorar el soporte para 64 bits. Más allá de esto, los cambios radicales al ensamblador no se han visto.

CÓDIGO

Macro que se utilizó para comparar las cadenas del user.

```
VerificarUsuario macro bufferUsuarios , usuario, mensaje1 , mensaje2
LOCAL INICIO , AVANZAR ,FIN , ERRORUSUARIO , CORRECTO
    xor si , si
    xor di , di
    xor ax , ax
    mov bl , 48                         ; Inicializo el valor de bl con 0 para decir que es falso
INICIO:
    mov al, bufferUsuarios[si]
    mov ah, usuario[di]
    cmp  al ,ah
    jne ERRORUSUARIO
AVANZAR:
    cmp al, 24h                         ; caracter de fin de linea
    je  FIN
    cmp ah, 3Bh
    je CORRECTO
    inc si
    inc di
    jmp INICIO
ERRORUSUARIO:
    xor di , di
    mov al, bufferUsuarios[si]
    inc si
    cmp al , 3Bh                          ; Espero a encontrar el salto de linea entre cada usuario en el archivo
    je  ERRORUSUARIO
    cmp al , 13                           ; Espero a encontrar el salto de linea entre cada usuario en el archivo
    je  ERRORUSUARIO
    cmp al ,  24h
    je  FIN
    cmp al ,  0Ah
    je  INICIO
    jmp ERRORUSUARIO

CORRECTO:
    mov bl, 49                    ; Bandera en bl es verdadera  = 1

FIN :


endm
```

Macro que se utilizo para pintar la pista del carrito

```
;------------------------------------------------------------ Pintar carretera --------------------
PintarCarretera macro buffer
LOCAL PINTAR, FIN , MOVERPUNTERO
            ; f(20,40) = x + y*320
            mov es, buffer              ;put segment address in es
            mov di, 9600                ; 30 * 320 = 9600
            add di, 15                  ; x = 15
            mov cx, 290                 ; loop counter
            mov ax, 07h                 ; color gris claro pra el cuadro

            PINTAR:
                mov es:[di], ax         ;set pixel to colour
                inc di                  ;move to next pixel
                cmp di , 59185          ;Final del cuadro
                je  FIN
            loop PINTAR

            MOVERPUNTERO:
                add di, 30              ; se suman los bordes del cuadro 15 + 15
                mov cx, 290             ; se inicia el contador con 290 para realizar el loop
                jmp PINTAR

            FIN :

endm
```

Macro para pintar una cadena en un lugar exacto de pantalla

```
;================================================================ Print Video Text in a specific place

Print_VideoString macro cadena , pagina , fila  , columna

    push    dx
    push    ax
    push    cx
    mov     ah, 2                       ; modo de escritura con el puntero en modo video
    mov     bh, pagina                  ; numero de pagina
    mov     dh, fila                    ; numero de fila
    mov     dl, columna                 ; numero columna
    int     10h                         ; ejecucion de la interrupcion

    mov     dx, offset cadena           ; tamañ de la cadena
    mov     ax, dow
    mov     cl, 0ah
    mul     cl
    mov     ah, 00h
    add     ax, dx
    mov     dx, ax

    mov     ah, 09h
    int     21h

    pop     cx
    pop     ax
    pop     dx

endm
```

Macro para pintar los bonos y obstáculos

```
;=========================================================== Pintar Estrella Mala ====================
Pintar_Obstaculo macro  startaddr , estrella_X , estrella_Y, estrella1 ,estrella2
LOCAL repetir , repetir2 , repetir3

                push ax
                push cx
                push dx

                mov es, startaddr                                       ;colocar direccion de segmento de video
                ;f(158,98) = x + y*320
                mov ax ,  estrella_Y                                    ;y*320 = 98*320 = 31360
                mov cx ,  320
                mul cx
                mov di , ax
                add di ,  estrella_X                                    ;sumar x
                mov dx ,          6                                     ; contador de alto de los bloques
                xor si ,         si
                mov cx ,         15                                     ;Ancho del bloque


                repetir:
                mov ax      , [estrella1+si]
                mov es:[di] ,  ax
                inc bx

                add si      , 2;
                inc di
                dec cx
                jnz repetir

                dec dx
                xor si      ,  si
                mov cx      ,  15                                   ;tamaño del dato a mover
                add di      , 305
                cmp dx      ,   0
                jg  repetir


                mov dx ,     8                                  ; contador de alto de los bloques
                xor si ,    si
                mov cx ,    15                                  ;Ancho del bloque

                repetir2:
```

Macro para pintar el carrito que recibe como parametro la posicion en X y la posicion Y con su color

```
;------------------------------------------------------------- Pintar Carro Moviendo--------------------
PintarCarroMoviendo macro buffer , xFinal , yPosition , colorHexadecimal
LOCAL PINTAR, FIN , MOVERPUNTERO
            ;buffer = la direccion de memoria VRAM
            ;xFinal = posicion Final del cursos para pintar el cuadrado
            ; f(x,y) = x + y*320
            mov es, buffer              ;put segment address in es
            mov di, yPosition           ; 125 * 320 = 40000
            add di, 16                  ; x = 16
            mov cx, 30                  ; loop counter =  el ancho de pixeles pintados
            mov bx, colorHexadecimal    ; color amarillo para el carro

            PINTAR:
                mov es:[di], bx         ;set pixel to colour
                inc di                  ;move to next pixel
                cmp di , xFinal         ; x = 46 ;  y = 183*320  -> Final 59246 del cuadro
                je  FIN
            loop PINTAR

            MOVERPUNTERO:
                add di, 290             ; se suman los bordes del cuadro 16 + 274 = 290
                mov cx, 30              ; loop counter = ancho del lo que se esta pintando en pixeles
                jmp PINTAR

            FIN :
endm

;=================================================================
```

MODO VIDEO UTILIZADO

Int 10h

INT 10h es la forma abreviada de la interrupción 0x10. Esta interrupción controla los servicios de pantalla del PC.

Realizando una unica llamada de la interrupcion para luego utilizar la memoria de video VRAM

realizando un un mapeo lexicografico de las posiciones para pintar en ellas . Lo cual hace mas rapida la ejecucion del codigo .

INTERRUPCIONES UTILIZADAS

Int 10h

INT 10h es la forma abreviada de la interrupción 0x10. Esta interrupción controla los servicios de pantalla del PC.

Int 21h

Esta interrupción tiene varias funciones, para accesar a cada una de ellas es necesario que el el registro AH se encuentre el número de función que se requiera al momento de llamar a la interrupción.

int 16h

INT 16h es la forma abreviada de la interrupción 0x16. Esta interrupción se encarga de controlar el teclado del PC.

Esta interrupción se encarga de obtener funcionalidades básicas del teclado, es decir, se encarga de recoger las pulsaciones del teclado, obtener el estado del buffer del teclado, etc. La codificación estándar del teclado que ofrece la INT 16h es de un teclado Estadounidense. Para adaptar la codificación de la INT 16h a otro tipo de teclado (por ejemplo, un teclado español) hay que atender al scan-code de la tecla pulsada y realizar las operaciones convenientes para interpretar la tecla que se desea.

En los teclados de 101 letras o más, existen unas teclas que la INT 16h las interpreta como teclas expandidas, que tienen un scan-code distinto al de las teclas normales (por ejemplo, la tecla pausa).