Laboratorio de arquitectura de computadores y ensambladores 1, Sección N.

PROYECTO 2

MANUAL TECNICO

FECHA: 01/07/2023

Juan Josue Zuleta Beb
Carné: 202006353

Introducción

El presente documento describe los aspectos técnicos informáticos de la aplicación de sistema de juego con modo video en ensamblador, deseñada a través del ensamblador masm611. El documento familiariza al personal técnico especializado encargado de las actividades de mantenimiento, revisión, solución de problemas, instalación y configuración del sistema.

Objetivos

Instruir el uso adecuado del de la instalación y comprensión del código y de la implementación de métodos, para el acceso oportuno y adecuado en la inicialización de este, mostrando los pasos a seguir en el proceso de inicialización, así como la descripción de los archivos relevantes del sistema los cuales nos orienten en la configuración.

Requisitos Mínimos del Sistema

Sistema operativo 64 bits

- Microsoft Windows 10/8/7/Vista/2003/XP (incl.64-bit)
- macOS 10.5 o superior
- Linux GNOME o KDE desktop
- Procesador a 1.6 GHz o superior
- 1 GB (32 bits) o 2 GB (64 bits) de RAM (agregue 512 MB al host si se ejecuta en una máquina virtual)
- 3 GB de espacio disponible en el disco duro
- Disco duro de 5400 RPM
- Tarjeta de vídeo compatible con DirectX 9 con resolución de pantalla de 1024 x 768 o más.
- Navegador web (Recomendado: Google Chrome)

Software (Indispensable tenerlo instalado)

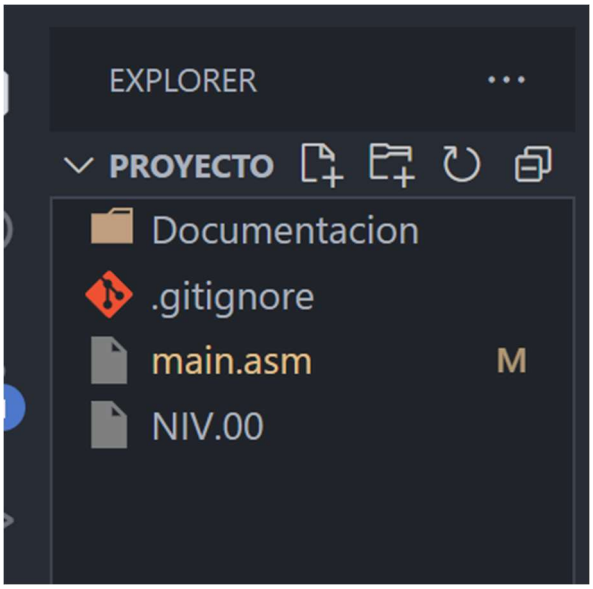
- IDE: Visual Studio Code
  Puede conseguirse en:
  https://code.visualstudio.com/

- MS-DOS (DOS-BOX)
  Puede conseguirse en:
  https://www.dosbox.com/download.php?main=1

- MASM611
  Puede conseguirse en:
  https://sourceforge.net/projects/masm611/

Estructura raíz:

El proyecto tiene la siguiente estructura de directorios:



Directorio de la App: Raiz

El archivo main.asm:

Contiene el método principal de nuestra aplicación y controlara todos los métodos y la implementación de la interfaz.

Los métodos y procedimientos utilizados son los siguientes:

Inicio:

El método inicial en su primera interacción activa el modo video con la interrupción 10

```
mov AH, 00
mov AL, 13
int 10
```

Como consiguiente se procede a escribir los datos del curso y del desarrollador en pantalla para lo cual se mandan a llamar variables de texto que se posicionan en pantalla por medio de la interrupción int 10

```
mis_datos_uni     db "UNIVERSIDAD DE SAN CARLOS DE GUATEMALA$"
mis_datos_fac     db "FACULTAD DE INGENIERIA$"
mis_datos_esc     db "ESCUELA DE VACACIONES$"
mis_datos_cur     db "LAB. ARQUITECTURA DE COMPUTADORES$"
mis_datos_cur2    db "Y ENSAMBLADORES 1$"
mis_datos_sec     db "SECCION N$"

mis_datos_nom     db "JUAN JOSUE ZULETA BEB$"
mis_datos_car     db "202006353$"
```

Seguido de esto se manda a llamar el método delay el cual genera un retardo en la reacción del programa para que el usuario tenga tiempo de leer los datos mostrados en pantalla.

```
delay:
        push SI
        push DI
        mov SI, 0200
```

Seguido de esto se manda a llamar al método inicio y se ejecuta el menú principal, el cual muestra las opciones del juego las cuales se seleccionan por medio de la tecla F1 y por medio de comparaciones de la posición actual de la flecha.

```
iniciar_juego db "INICIAR JUEGO$"
cargar_nivel  db "CARGAR NIVEL$"
configuracion db "CONFIGURACION$"
puntajes      db "PUNTAJES ALTOS$"
salir         db "SALIR$"
```

Las opciones mostradas se describen como sigue, en orden en el que se muestran:

Iniciar Juego:

Para este método se realiza una búsqueda de archivos con nombres específicos dentro de la carpeta raíz, de no existir deberá lanzar un error, y al ser encontrados redirigirá al método de generación de mapas

```
cargar_un_nivel_auto:
    cmp [sesion_activa], 01
    je ciclo_juego
    mov cont_sent_jug , 0
    mov cont_sent_caj , 0
    mov cont_sent_obj , 0

    mov AL, 00
    mov DX, offset nivel_cero
    mov AH, 3d
    int 21
    jc archivo_no_encontrado
    mov [handle_nivel], AX
    jmp ciclo_lineas
```

Para generar el mapa se parsean las entradas de texto contenidas en el archivo y se encarga de asginarlos a las variables globales de caracteres de jugadores, suelo, paredes y objetivos, así como de cantidad de los mismos.

```
ciclo_lineas:
        mov BX, [handle_nivel]
        call siguiente_linea
        cmp DL, 0ff
        je fin_parseo
        cmp DH, 00
        je ciclo_lineas

        mov DI, offset linea
        push DI
        mov SI, offset tk_pared
        call cadena_igual
        cmp DL, 0ff
```

Una vez se ha terminado de leer todo el archivo y el parseo de los datos se procede a compara si los datos que ingresan son validos debido a que debe haber como mínimo una sentencia de jugador y la misma cantidad de cajas que de objetivos para que el juego pueda ser manipulable o tenga propósito por lo cual antes de iniciar el juego se realiza la validación de los datos de entrada comparados con contadores globales y así poder corroborar la existencia y la validez de los mismos.

```
fin_parseo:
        ;; cerrar archivo
        mov AH, 3e
        mov BX, [handle_nivel]
        int 21
        ;;
        int 03

        mov AH, cont_sent_jug
        cmp AH, 01
        jne archivo_no_encontrado2

        mov AH, cont_sent_caj
        cmp AH, cont_sent_obj
        jne archivo_no_encontrado3

        jmp ciclo_juego
        jmp fin
```

Si el archivo contiene alguno de estos errores o no cumple con requisitos mínimos se lanzaran los mensajes de error correspondientes y se regresara al menú inicial para que el usuario pueda volver a intentar o cambiar sus archivos de entrada.

Cargar un nivel:

Para la opción de cargar un nivel se le requiere al usuario ingresar el nombre del archivo que desea buscar, por medio de la interrupción int 21

Esto se recibe a través de un buffer de entrada de como máximo 20 caracteres, una vez recibida la cadena se procede a analizar los caracteres de la misma y comparar si existe en el directorio raíz dicho archivo. En caso que no exista lanzara un error y redirigirá al usuario al menú principal para que pueda volver a ingresar el nombre de un archivo o que pueda

asegurarse de que el archivo que desea cargar contenga los datos correctos.

```
cargar_un_nivel:
        call clear_map_buffer
        call clear_pantalla
        mov DL, 0b
        mov DL, 1
        mov DH, 03
        mov BH, 00
        mov AH, 02
        int 10

        push DX
        mov DX, offset ask_filename
        mov AH, 09
        int 21
        pop DX
```

Una vez finalizado este método se procede a realizar la misma validación que en el método anterior por lo cual si el archivo no cumple con requisitos mínimos en la validación de datos no podrá ser cargado y lanzara los mensajes de error correspondientes a la falla.

Puntajes altos:

Método no implementado, debería recolectar los datos en partidas donde el jugador ha logrado colocar cada caja sobre cada objetivo y realizar un top de los 10 mejores puntajes durante los juegos, estos serian guardados en un archivo binario y almacenados de forma serial para la lectura y perduración de datos aun cuando la aplicación finalice su ejecución.

Configuración:

Para el menú de configuración se despliega una serie de cadenas las cuales nos indican la configuración actual de los controles, estas nos ayudan a poder reflejar los controles actuales y los nuevos controles una vez hayan sido configuración de la opción de cambiar controles.

```
;;CONFIGURACION
configuracion_titulo db "CONTROLES ACTUALES$"
configuracion_titulo2 db "INGRESAR NUEVOS CONTROLES$"
configuracion_arriba db "ARRIBA:$"
configuracion_abajo db "ABAJO:$"
configuracion_izquierda db "IZQUIERDA:$"
configuracion_derecha db "DERECHA:$"

configuracion_guardar db "CAMBIAR CONTROLES$"
configuracion_salir db "REGRESAR$"
```

Al ingresar a la configuración de controles nuevos se desplegaran de forma secuencial la petición de nuevos controles, por medio de la interrupción int 16, esta devolverá la tecla actual presionada en el registro AH, y guardara el carácter presionado en el registro AL.

```
mov DI, offset configuracion_nueva_arriba
mov Cx , 0d
call clear_buffer

mov AH, 00
int 16

mov [control_arriba], AH
mov configuracion_nueva_arriba[0], AL
```

De esta forma se sustituyen los valores actuales del control en las variables globales y así mismo, en las etiquetas utilizadas para desplegar la información en el menú de configuración.

Opción salir:

Llamara automáticamente la función .exit lo cual terminara con el programa de forma abrupta, los valores almacenados en archivos binarios permanecerán para la nueva ejecución, todos aquellos valores no almacenados se perderán.