UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERIA

ESCUELA DE SISTEMAS

ARQUITECTURA DE COMPUTADORAS Y ENSAMBLADORES 1

ING. ÁLVARO OBRAYAN HERNÁNDEZ GARCÍA

AUX. RONALD MARÍN

PROYECTO 2

SOKOBAN

JOSUE ROLANDO

GRAMAJO ROLDAN

202000895

3021021080101

INDICE

Introducción

Para la presente actividad se le solicita el desarrollo de un juego sencillo empleando las características gráficas que brinda DOS y el conjunto de interrupciones que este provee. Junto a este juego, se desarrollará la interfaz que permitirá manejarlo, el manejo de puntajes más altos, carga de niveles y configuración de controles.

El juego por desarrollar es el juego japonés **Sokoban**, en el cual el jugador tiene como principal objetivo empujar una serie de cajas hasta conseguir que éstas se ubiquen en ciertas posiciones.

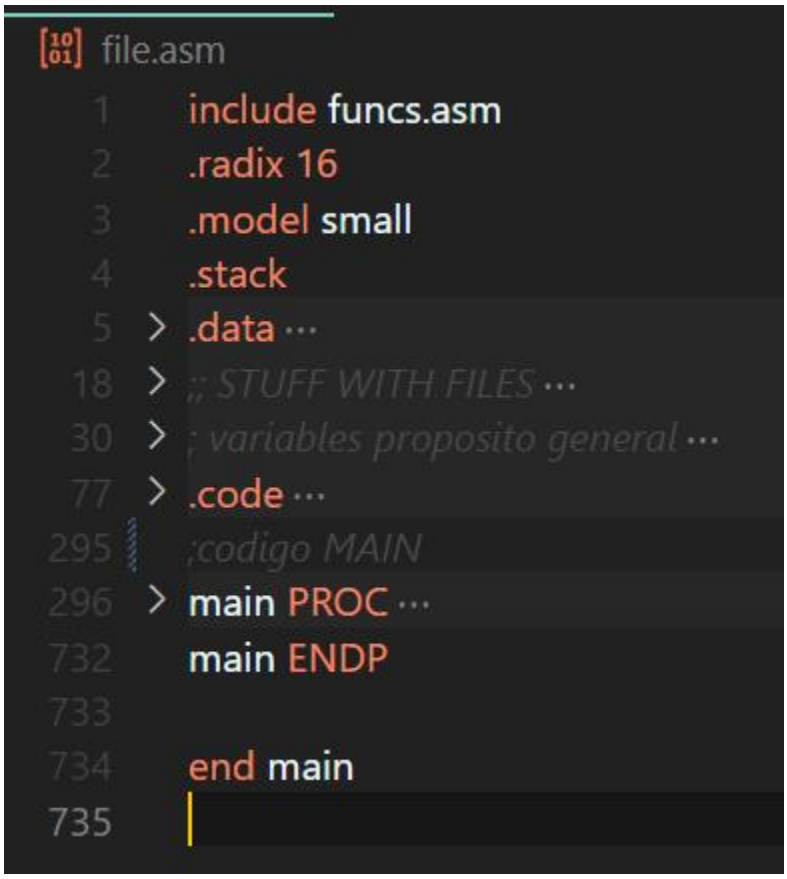
Objetivos

Guiar al lector en la manipulación del programa, para un uso correcto y fluido, de esa manera se evitarán bugs o posibles errores en la aplicación.

Requerimientos mínimos para ejecutar en DOSBOX

- Sistema operativo: Windows XP o posterior
- Procesador: Pentium 200 MHz o superior
- Memoria RAM: 64 MB de RAM
- Espacio en disco: Alrededor de 10 MB para la instalación
- Tarjeta de sonido compatible con Windows
- Tarjeta gráfica compatible con DirectX 7 o superior

Descripción de código



Estructura general del archivo .asm

.include Como en cualquier lenguaje de alto nivel podemos llamar a otros archivos que contengan código para reutilizar en nuestro programa.

.radix 16 Significa que estaremos usando el sistema Hexadecimal.

.data Es el apartado donde se definen las variables. Se pueden crear variables con diferentes tamaños aunque los casos mas utilizados son los de cadenas tipo ***db*** o ***dw.***

```
.DATA
dim_sprite_jug    db  08, 08
data_sprite_jug   db  5c, 5c, 5c, 04, 04, 04, 5c, 5c
                  db  5c, 5c, 5c, 54, 54, 5c, 5c, 5c
                  db  5c, 5c, 02, 02, 02, 02, 5c, 5c
                  db  5c, 54, 5c, 02, 02, 5c, 54, 5c
                  db  5c, 5c, 5c, 09, 09, 5c, 5c, 5c
                  db  5c, 5c, 5c, 09, 09, 5c, 5c, 5c
                  db  5c, 5c, 54, 5c, 5c, 54, 5c, 5c
                  db  5c, 00, 00, 5c, 5c, 00, 00, 5c
dim_sprite_flcha  db  08, 08
data_sprite_flcha  db  00, 00, 8F, 00, 00, 00, 00, 00
                   db  00, 00, 8F, 8F, 00, 00, 00, 00
                   db  8F, 8F, 8F, 8F, 8F, 00, 00, 00
                   db  8F, 8F, 8F, 8F, 8F, 8F, 00, 00
                   db  8F, 8F, 8F, 8F, 8F, 8F, 00, 00
                   db  8F, 8F, 8F, 8F, 8F, 00, 00, 00
                   db  00, 00, 8F, 8F, 00, 00, 00, 00
                   db  00, 00, 8F, 00, 00, 00, 00, 00
dim_sprite_vacio  db  08, 08
data_sprite_vacio db  00, 00, 00, 00, 00, 00, 00, 00 ···
dim_sprite_suelo  db  08, 08
data_sprite_suelo db  5c, 5c, 5c, 5c, 5c, 5c, 5c, 5c
                  db  5c, 5c, 5c, 5c, 5c, 5c, 5c, 5c
                  db  5c, 5c, 5c, 5c, 5c, 5c, 5c, 5c
                  db  5c, 5c, 5c, 5c, 5c, 5c, 5c, 5c
                  db  5c, 5c, 5c, 5c, 5c, 5c, 5c, 5c
                  db  5c, 5c, 5c, 5c, 5c, 5c, 5c, 5c
                  db  5c, 5c, 5c, 5c, 5c, 5c, 5c, 5c
                  db  5c, 5c, 5c, 5c, 5c, 5c, 5c, 5c
dim_sprite_pared  db  08, 08
data_sprite_pared db  6,6,0,0,0,6,6,6 ···
dim_sprite_caja   db  08, 08
data_sprite_caja  db 5c,5c,5c,5c,5c,5c,5c,5c ···
dim_sprite_obj    db  08, 08
data_sprite_obj   db 0,2A,2A,2A,2A,2A,2A,2A ···
dim_sprite_mario    db  08, 08
data_sprite_mario   db 5C,28,28,28,28,5C,5C,5C ···
```

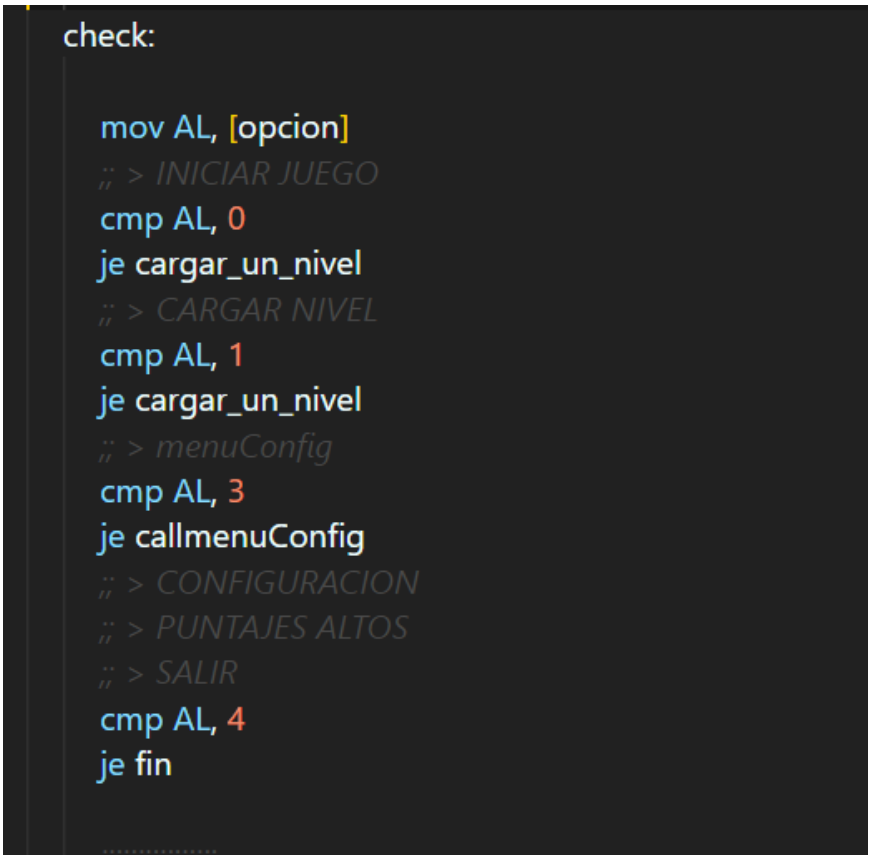
```
stringcontrol_abajo     db "$"
stringcontrol_derecha     db "$"
stringcontrol_izquierda     db "$"
longitud_cadena equ $-stringcontrol_izquierda
stringcontrol_arriba     db "$" ···
flagHayX db 0
contNivelitos db 0 ···
control_arriba    db  48
control_abajo     db  50
control_izquierda db  4b
control_derecha   db  4d
```

Estas variables son para el manejo de todo el juego, tenemos SPRITE que representan figuras en 2D y variables de uso general.

```
.CODE
.STARTUP
inicio:
    ;; MODO VIDEO ;; ···
    check: ···
ciclo_juego: ···
cargar_un_nivel: ···
ciclo_lineas: ···
es_pared: ···
es_caja: ···
es_suelo: ···
es_objetivo: ···
es_jugador: ···
continuar_parseo0: ···
continuar_parseo1: ···
continuar_parseo2: ···
ver_final_de_linea: ···
guardar_coordenadas_jugador: ···
fin_parseo: ···
```

Inicio Esta etiqueta cargamos el segmento de DATA y definimos el modo VIDEO.

El flujo continua parseando el archivo que contiene el mapa del juego y se carga al sistema para proceder a mostrar el menú principal.



```
check:

  mov AL, [opcion]
  ;; > INICIAR JUEGO
  cmp AL, 0
  je cargar_un_nivel
  ;; > CARGAR NIVEL
  cmp AL, 1
  je cargar_un_nivel
  ;; > menuConfig
  cmp AL, 3
  je callmenuConfig
  ;; > CONFIGURACION
  ;; > PUNTAJES ALTOS
  ;; > SALIR
  cmp AL, 4
  je fin

  ...............
```

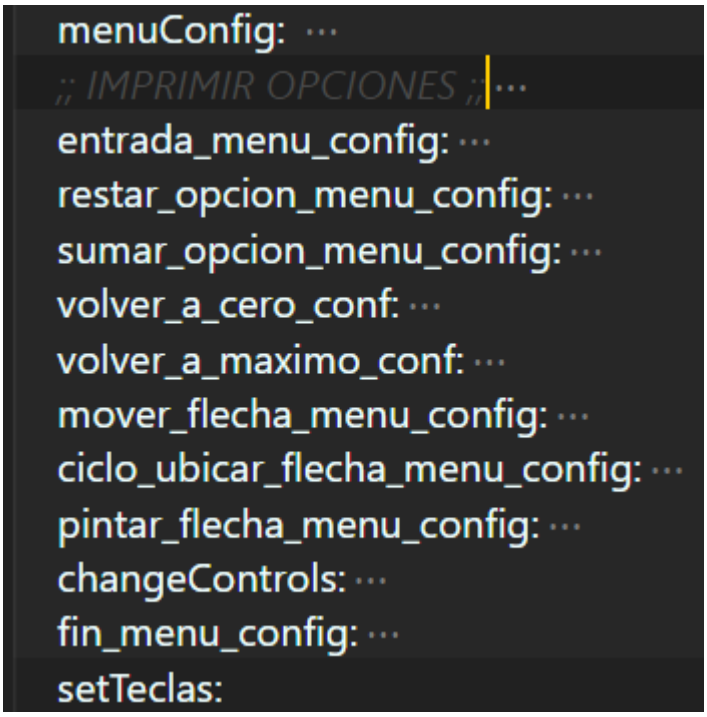
check Sera nuestra etiqueta que decidirá el flujo del programa después de imprimir el menú de opciones.

```
;; pintar_mapa - pinta los elementos del mapa
;; ENTRADA:
;; SALIDA:
pintar_mapa: ···
ciclo_v: ···
ciclo_h: ···
pintar_vacio_mapa: ···
pintar_suelo_mapa: ···
pintar_jugador_mapa: ···
pintar_pared_mapa: ···
pintar_caja_mapa: ···
pintar_objetivo_mapa: ···
continuar_h: ···
continuar_v: ···
fin_pintar_mapa: ···
```

pintarMapa Esta etiqueta es la que se encarga de pintar todo el mapa con sus componentes como Pared, Suelo, Objetivos y Jugador.

```
entrada_menu_pause:
    mov AH, 00
    int 16
    cmp AH, 48
    je restar_opcion_menu_pause
    cmp AH, 50
    je sumar_opcion_menu_pause
    cmp AH, 3b  ;; le doy F1
    je fin_menu_pause
    jmp entrada_menu_pause
restar_opcion_menu_pause: ···
sumar_opcion_menu_pause: ···
volver_a_cero_pause: ···
volver_a_maximo_pause: ···
mover_flecha_menu_pause: ···
ciclo_ubicar_flecha_menu_pause: ···
pintar_flecha_menu_pause: ···
fin_menu_pause:
    ; mov [opcion], 00
    cmp opcion, 0
    je ciclo_juego
    cmp opcion, 1
    je sendMenu
    ; ret
```

menu_pause Para detener la ejecución del juego podemos acceder al menú de pausa presionando la tecla F2, nos despliega opciones para continuar o salir del juego y regresar al menú principal.



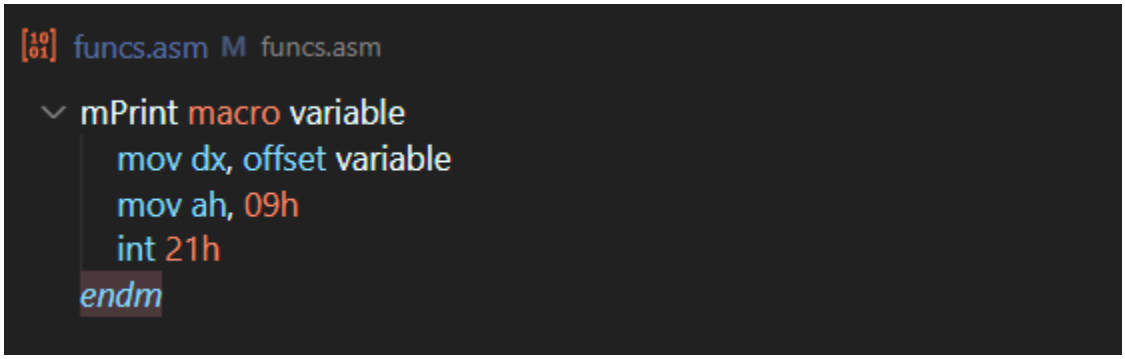
menuConfig El juego nos permite modificar las teclas DEFAULT de juego a nuestro gusto, simplemente presionando la tecla que deseemos.

```
lvls:
    cmp contNivelitos, 0
    je lvlone
    cmp contNivelitos, 1
    je lvltwo
    cmp contNivelitos, 2
    je cleanGame
lvlone:
    inc contNivelitos
    mov contadorWin, 0000
    mov DI , offset mapa
    mov CX, 3e8
    mov AL, 00000
    call memset
    mov nivel_x[7], "1"
    jmp cargar_un_nivel
lvltwo: ...
cleanGame:
    mov contadorWin, 0000
    mov contNivelitos, 0000
    jmp menu_principal
```

Lvls Esta etiqueta compara el nivel que acaba de ganar el jugador y dependiendo de su valor redirige al siguiente nivel, si en dado caso el valor se pasa del total de niveles regresamos al menuPrincipal.

Limpiamos todas las variables para continuar con un flujo nuevo.

Archivo funcs.asm



mPrint Este macro lo utilizamos para imprimir lo que deseemos.

Ejemplo archivo de entrada NIV.00

pared 00, 00
pared 01, 00
pared 02, 00
pared 03, 00
pared 04, 00
pared 05, 00
-
pared 05, 00
pared 05, 01
pared 05, 02
pared 05, 03
pared 05, 04
pared 05, 05
-
pared 01, 05
pared 02, 05
pared 03, 05
pared 04, 05
pared 05, 05
-
pared 00, 01
pared 00, 02
pared 00, 03
pared 00, 04
pared 00, 05

suelo 01, 01
suelo 01, 02
suelo 01, 03
suelo 01, 04

suelo 02, 01
suelo 02, 02
suelo 02, 03
suelo 02, 04

suelo 03, 01
suelo 03, 02
suelo 03, 03
suelo 03, 04

suelo 04, 01
suelo 04, 02
suelo 04, 03
suelo 04, 04
objetivo 02, 02

caja 02, 03

jugador 01, 01