Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de ciencias y sistemas

Arquitectura de computadores y ensambladores 1

Guía Técnica

Herberth Guillermo Obregon Espino

Carne 201314237

Guatemala 21 de Abril del 2017

Introducción

Se realizó el popular juego de snake. en el cual se contará con niveles y también se tendrá la opción de poder registrarse e ingresar al juego, esto con la finalidad de que se pueda generar el reporte de Top 10, por puntos obtenidos y por el mayor tiempo de juego. Al momento de iniciar la aplicación tendrá que permitir el ingreso o registro de un usuario, entonces se desplegará un menú principal que contendrá las siguientes opciones:

− Ingresar  
 − Registrar  
− Salir

Modo video utilizado

mov ah,00h ;modo video

mov al,13h ;320x200 256 colores

int 10h ;servicio de pantalla

Métodos

; Universidad de San Carlos de Guatemala

; Herberth Guillermo Obregon Espino

; Arquitectura de computadores y ensambladores 1

; Seccion B

;|===================================================================|

;| F U N C T I O N S |

;|===================================================================|

%macro write\_archivo 3

mov dx, %1 ; prepara la ruta del archivo

mov ah, 3ch ; funcion 3c, crear un archivo

mov cx, 0 ; crear un archivo normal

int 21h ; interrupcion DOS

mov bx, ax ; se guarda el puntero del archivo retornado de la funcion

mov ah, 40h ; funcion 40, escribir un archivo

mov cl, %2

mov dx, %3 ; preparacion del texto a escribir

int 21h ; interrupcion DOS

mov ah, 3eh ; funcion 3e, cerrar un archivo

int 21h ; interrupcion DOS

%endmacro

%macro println 1

mov dx, %1 ; coloca direccion de cadena en dx

mov ah,09h ; funcion 09h, imprimir en pantalla lo que esta en dx

int 21h ; interrupcion DOS

%endmacro

%macro printlnWait 1

println mEnter

mov dx, %1 ; coloca direccion de cadena en dx

mov ah,09h ; funcion 09h, imprimir en pantalla lo que esta en dx

int 21h ; interrupcion DOS

call readline

%endmacro

;%macro readBytesFile 1

;Lectura de Fichero o dispositivo

;LLAMADA:

;AH = 3FH

;BX = Handle.

;CX = Número de bytes a leer.

;DS:DX = Segmento: Desplazamiento del buffer donde se depositarán los caracteres leídos

;DEVUELVE:

;Si se ejecutó correctamente: Flag de acarreo (Cf) = 0

;AX = Bytes transferidos

;Si NO se ejecutó correctamente: Flag de acarreo (Cf) = 1

;AX = Código de error.

; xor ah,ah

; xor dx,dx

; xor bx,bx

; xor cx,cx

; mov ah, 3Fh ; Lectura de fichero

; mov bx, [Handler] ; Le mando el handler del fichero

; mov cx, %1 ; Numero\_bytes por leer

; mov dx, mBufferNameReg

; int 21h ; interrupcion DOS

; jc Salir\_Error

;%endmacro

ORG 100h

section .text

global \_start

\_start:

call cls

println mHeader ; Imprime el Header

printlnWait mMenu ; Imprime el Header y espera el input

call comparar\_entrada

ret

\_startAdmin:

call cls

println mHeader ; Imprime el Header

printlnWait mMenuAdmin ; Imprime el Header y espera el input

call comparar\_entradaa

ret

cls:

mov ah,00h ;modo video

mov al,03h ;03 Video Normal

int 10h ;servicio de pantalla

ret

error\_usuario\_NoPermitido:

println mEnter

println mUsuarioNoPermitido

call readline

ret

error\_pass\_NoPermitido:

println mEnter

println mPassNoPermitido

call readline

ret

readline:

;(si AH =08H) es servicio sin Echo)

;AH = 01H es servicio con Echo)

mov ah,08h ; Capturar la entrada del teclado

int 21h ; interrupcion DOS

ret

readlineEcho:

;mov ah,08h es servicio sin Echo)

;mov ah,01h es servicio con Echo)

;return

;al = Código ASCII del Carácter leído y Echo a pantalla

mov ah,01h ; Capturar la entrada del teclado

int 21h ; interrupcion DOS

ret

Salir:

call cls

mov ah,04ch ;termina el programa

int 21h ;llama el kernel para realizar la acción

ret

oArchUsuarios:

mov word[Handler], 0

mov ah,3dh ;Para Abrir un Archivo

mov al,2h ;0h solo lectura, 1h solo escritura, 2 lectura y escritura

mov dx, userstxt

int 21h ; interrupcion DOS

mov [Handler], ax

ret

oArchReportes:

mov word[Handler], 0

mov ah,3dh ;Para Abrir un Archivo

mov al,2h ;0h solo lectura, 1h solo escritura, 2 lectura y escritura

mov dx, reportestxt

int 21h ; interrupcion DOS

mov [Handler], ax

ret

Escribir\_Archivo:

mov ah,40h ; Escritura en Fichero o dispositivo.

mov bx,[Handler]

int 21h ; interrupcion DOS

ret

Seek\_End:

mov ah, 42h ; mov file pointer

mov al, 2 ; set to end

mov bx, [Handler] ; Le mando el handler del fichero

mov cx, 0 ; desplazamiento

mov dx, 0 ; Buffer

int 21h ; interrupcion DOS

ret

Obtener\_Numero\_Bytes:

mov word [Numero\_bytes], 0

ContadordeCaracters:

mov al, [si]

mov bl, 0

cmp al, bl

je Salida\_NumeroBytes

add si, 1

inc word [Numero\_bytes]

jmp ContadordeCaracters

Salida\_NumeroBytes:

ret

Error\_Cerrar\_Archivo:

printlnWait mErrorCerrarArchivo

ret

Usuario\_Duplicado:

printlnWait errorUsarioDuplicado

call Close\_File

mov dx,00h

mov [Handler],dx

jmp \_start

ret

Solo\_Numeros:

printlnWait mErrorSoloNumPass

ret

Close\_File:

call Seek\_End ; Se va al final del archivo para comenzar a escribir ;envia el puntero al final del archivo

mov bx,[Handler] ;variable que contiene el abrir\_archivo

mov ah,3eh ;cierra el archivo

int 21h ; interrupcion DOS

jc Error\_Cerrar\_Archivo

ret

Salir\_Error:

printlnWait MsgError

jmp \_start

ret

LimpiarVariables:

mov word [Handler], 00h

mov di, 0 ; Inicializa cont = 0

RegresarUsuario:

mov si, 0

mov si, mUserLogin

add si, di

mov byte[si], ' '

mov si, 0

mov si, mUserRegistro

add si, di

mov byte[si], ' '

mov si, 0

mov si, mUsuario

add si, di

mov byte[si], ' '

mov si, 0

mov si, mBufferNameReg

add si, di

mov byte[si], ' '

inc di ; contador++

cmp di, 5

jl RegresarUsuario

mov byte [si], '$'

mov di, 0 ; Inicializa cont = 0

RegresarPassword:

mov si, 0

mov si, mPasswordLogin

add si, di

mov byte[si], ' '

mov si, 0

mov si, mPasswordRegistro

add si, di

mov byte[si], ' '

mov si, 0

mov si, mPasswordR

add si, di

mov byte[si], ' '

inc di ; contador++

cmp di, 5

jl RegresarPassword

mov byte [si], '$'

ret

;======================================================================|

; D A T A |

;======================================================================|

section .data

mHeader db "|=====================================================|",0,0ah,"| Universidad de San Carlos de Guatemala |",0,0ah,"| Faculta de Ingenieria |",0,0ah,"| Arquitectura de Computadores y Ensambladores 1 |",0,0ah,"| Seccion B |",0,0ah, "| Herberth Guillermo Obregon Espino |",0,0ah,"| 201314237 |",0,0ah,"|=====================================================|",0,0ah,"$"

mMenu db 0ah,0ah,9h,"1. Ingresar",0ah,0dh,9h,"2. Registrarse",0ah,0dh,9h,"3. Salir",0ah,0dh,24h,"Eliga una opcion:",0ah,0dh,24h

mMenuAdmin db 0ah,0ah,9h,"1. Top 10 puntos",0ah,0dh,9h,"2. Top 10 tiempos",0ah,0dh,9h,"3. Salir",0ah,0dh,24h,"Eliga una opcion:",0ah,0dh,24h

mEnter db 0Ah,'$'

mTopTiempo: db " Top 10 Tiempos $",0

mTopPts: db " Top 10 puntos $",0

userstxt: db "USUARIOS.TXT", 0

reportestxt: db "REPORTES.TXT", 0

; Registro de usuario

mMsgRegistro db 9h,9h,"Registro de un nuevo Usuario",0ah,0dh,24h

mIngreseUsuario: db "Ingrese el nombre de usuario: $"

mUsuarioNoPermitido: db "Usuario no Permitido$", 0ah,0dh,24h

mIngresePassword: db "Ingrese su contrasenia:$ "

mErrorSoloNumPass db "Solo se aceptan numeros$", 0ah,0dh,24h

mMsgExito db "Usuario registrado exitosamente! $",0ah,0dh,24h

mPassNoPermitido: db "Pass no Permitida$", 0ah,0dh,24h

mErrorCerrarArchivo: db "Ingese el nombre de usuario: $"

MsgError: db "Se ha producido un error$", 0ah,0dh,24h

errorUsarioDuplicado: db " Error Usuario Duplicado$", 0ah,0dh,24h

; Verificar ingreso de usuario

mMsgErrorUsuario: db "Usuairo Incorrecto$",0ah,0dh,24h

mMsgErrorPass: db "Password Incorrecto$",0ah,0dh,24h

mUserLogin: db " $", 0 ; Aqui se concatenan los char en un readline

mUserRegistro: db " ", 0 ; $ el que se escribe en un archivo

mUsuario: db " $", 0 ; Este sirve para compara cardenas porque tiene $

mPasswordLogin: db " $", 0 ; Aqui se concatenan los char en un readline

mPasswordRegistro: db " ", 0

Numero\_bytes: dw 00h

LineQ: db " $",0ah,0dh,24h

mBufferOneChar: db "$$", 0

mBufferTwoChar: db "$$$", 0

mBufferQChar: db "$$$$$$$$$$$$$$$$$$", 0

mPasswordR: db "$$$$$$", 0

mBufferNameReg: db "$$$$$$$$", 0

Handler: dd 00h

Caracter: db '0','$'

Coma: db ",",'$'

admin: db "adminbi",0ah,0dh,24h

adminpass: db "4321",0ah,0dh,24h

path: db "arch.dat",00h

usac: db "Universidad de San Carlos de Guatemala$",0ah,0dh,24h

usac2: db "Universidad de San Carlos de Guatemala ADMIN$",0ah,0dh,24h

usaclen: equ $ - usac

filename: db "./output.txt", 0

text: db "Universidad de san carlos$"

textlen: equ $ - text

strlen: db 3 ;;;(GIVE THE LENGTH OF THE SNAKE HERE)initial length should not be more than 12...

string: dw 0x0204,0x0304,0x0404,0x0504,0x0604,0x0704,0x0804,0x0904,0x0a04,0x0b04,0x0c04,0x0d04,0x0b04,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ; contains the x-coordinate followed by y-quardinate ;;

Food: dw 1000,2000,3000,1100,2700,1694,2500,28,2900 ;;

FoodPtr: dw 0 ;;

DiscardedTail: dw 0 ; not for the user, it is to lengthen the snake ;;

;;

StrPosition: db 0 ; not for the user ;;

direction: db 2 ; up, down, left or right direction needed for proper head selection ;;

SubDirection: db 0 ; for every individual byte, up, down, left or right for appropriate previous byte printing ;;

mmsgfinjuego: db "Chocaste con el cuerpo de la serpiente",0ah,0dh,24h

newthing: db 0x4d

username: db "Herberth",0ah,0dh,24h

punteo: db "Punteo:00", 0ah,0dh,24h

;======================================================================|

; M A I N |

;======================================================================|

verificar\_usuario:

call cls

println mHeader

call LimpiarVariables

println mIngreseUsuario

println mEnter

mov di, 0 ; Inicializa cont = 0

EscribirNombreLogin:

call readlineEcho

cmp al,0dh ; Compara si es "ENTER"

jne UnirUser

jmp SeguirUser

UnirUser:

mov si, 0

mov si, mUserLogin

add si, di

mov byte [si], al

mov si, 0

mov si, mUsuario

add si, di

mov byte [si], al

SeguirUser:

inc di ; contador++

cmp di, 9 ; Compara que sea < 8 (9 porque el \n cuenta como char)

je error\_usuario\_NoPermitido

cmp al,0dh ; Compara si es "ENTER"

jne EscribirNombreLogin

println mEnter

println mIngresePassword

println mEnter

mov di, 0 ; Inicializa cont = 0

EscribirPasswordLogin:

call readlineEcho

cmp al,0dh ; Compara si es "ENTER"

jne UnirPass

jmp SeguirPass

UnirPass:

mov si, 0

mov si, mPasswordLogin

add si, di

mov byte [si], al

SeguirPass:

inc di ; contador++

cmp di, 6 ; Compara que sea < 5 (6 porque el \n cuenta como char)

je error\_pass\_NoPermitido

cmp al,0dh ; Compara si es "ENTER"

jne EscribirPasswordLogin

call Leer\_Archivo\_Usuarios

ret

registrar\_usuario:

call cls

println mHeader

println mEnter

println mMsgRegistro

println mEnter

call LimpiarVariables

println mIngreseUsuario

println mEnter

mov di, 0 ; Inicializa cont = 0

; ======= AHORA PIDO EL USUARIO ============

wNombreRegistro:

call readlineEcho

cmp al,0dh ; Compara si es "ENTER"

jne concatUserChar ; else -> concatenar caracteres

jmp SeguirUserR ; IF(keypressed = ENTER) -> Seguir con registro Usuario

concatUserChar:

mov si, 0

mov si, mUserRegistro

add si, di

mov byte [si], al

mov si, 0

mov si, mUsuario

add si, di

mov byte [si], al

SeguirUserR:

inc di ; contador++

cmp di, 9 ; Compara que sea < 8 (9 porque el \n cuenta como char)

je error\_usuario\_NoPermitido

cmp al,0dh ; Compara si es "ENTER"

jne wNombreRegistro

call oArchUsuarios

EncontrarUsuarioR:

;Lectura de Usuario

xor ah,ah

xor dx,dx

xor bx,bx

xor cx,cx

mov ah, 3Fh ; Lectura de fichero

mov bx, [Handler] ; Le mando el handler del fichero

mov cx, 7 ; Numbytes x leer, 7 por el tamaño del username

mov dx, mBufferNameReg

int 21h ; interrupcion DOS

jc Salir\_Error

Comprobacion\_UsuarioR:

lea si, [mUsuario]

lea di, [mBufferNameReg]

repe cmpsb ; compara 2 str de si == di

jne AvanzarR

je Usuario\_Duplicado

AvanzarR:

xor ah,ah

xor dx,dx

xor bx,bx

xor cx,cx

mov ah, 3Fh ; Lectura de fichero

mov bx, [Handler] ; Le mando el handler del fichero

mov cx, 7 ; Leo otros 7 porque asi llego al sig username

int 21h ; interrupcion DOS

jc Salir\_Error

cmp ax, 0 ; IF (bytes trasferidos){ NADA.. } else -> jne

jne EncontrarUsuarioR

call Close\_File

; ====== AHORA PIDO LA CONTRASEÑA ============

println mEnter

println mIngresePassword

println mEnter

mov di, 0 ; Inicializa cont = 0

EscribirPasswordRegistro:

call readlineEcho

cmp al,39H ;compara con el ascii de 9 si se ingresa un ascii mayor a ese despliega el error

ja Solo\_Numeros ;se activa cuando el segundo es mayor

cmp al,0dh ; Compara si es "ENTER"

jne UnirPassR

jmp SeguirPassR

UnirPassR:

mov si, 0

mov si, mPasswordRegistro

add si, di

mov byte [si], al

SeguirPassR:

inc di ; contador++

cmp di, 6 ; Compara que sea < 5 (6 porque el \n cuenta como char)

je error\_pass\_NoPermitido

cmp al,0dh ; Compara si es "ENTER"

jne EscribirPasswordRegistro

Registrar:

call oArchUsuarios

xor ah,ah

xor dx,dx

xor bx,bx

xor cx,cx

call Seek\_End ; Se va al final del archivo para comenzar a escribir

; Escribo el nombre del usuario

mov si, mUserRegistro

call Obtener\_Numero\_Bytes

mov cx,[Numero\_bytes]

mov dx, mUserRegistro

call Escribir\_Archivo

; Escribo una coma

mov cx,1

mov dx,Coma

call Escribir\_Archivo

; Escribo la pass del usuario

mov si, mPasswordRegistro

call Obtener\_Numero\_Bytes

mov cx,[Numero\_bytes]

mov dx,mPasswordRegistro

call Escribir\_Archivo

; Escribo un enter del usuario

mov cx,1

mov dx,mEnter

call Escribir\_Archivo

call Close\_File

println mEnter

printlnWait mMsgExito

jmp \_start

ret

Leer\_Archivo\_Usuarios:

call oArchUsuarios

xor ah,ah

xor dx,dx

xor bx,bx

xor cx,cx

mov ah, 3Fh ; Lectura de fichero

mov bx, [Handler] ; Le mando el handler del fichero

mov cx, 7 ; Numbytes x leer, 7 por el tamaño del username

mov dx, mBufferNameReg

int 21h ; interrupcion DOS

jc Salir\_Error

lea si, [mUsuario]

lea di, [mBufferNameReg]

repe cmpsb ; compara 2 str de si == di

jne UserNormal

;Lectura de la "Coma"

xor ah,ah

xor dx,dx

xor bx,bx

xor cx,cx

mov ah, 3Fh ; Lectura de fichero

mov bx, [Handler] ; Le mando el handler del fichero

mov cx, 1 ; Me salto una coma

int 21h ; interrupcion DOS

jc Salir\_Error

;Lectura de Password

xor ah,ah

xor dx,dx

xor bx,bx

xor cx,cx

mov ah, 3Fh ; lectura de fichero

mov bx,[Handler] ; Le mando el handler del fichero

mov cx, 5 ; Leo la pass de 5 chars

mov dx, mPasswordR ; Le mando el buffer para guardar

int 21h ; interrupcion DOS

jc Salir\_Error

println mEnter

println mEnter

println mPasswordR

println mEnter

println mPasswordLogin

println mEnter

;Comprobacion\_Password:

lea si, [mPasswordLogin]

lea di, [mPasswordR]

repe cmpsb ; compara 2 str de si == di

je UserAdmin

UserAdmin:

call Close\_File

call cls

jmp \_startAdmin

UserNormal:

xor ah,ah

xor dx,dx

xor bx,bx

xor cx,cx

mov ah, 3Fh ; Lectura de fichero

mov bx, [Handler] ; Le mando el handler del fichero

mov cx, 7 ; Leo otros 7 para llegar al sig username

int 21h ; interrupcion DOS

jc Salir\_Error

EncontrarUsuario:

;Lectura de Usuario

xor ah,ah

xor dx,dx

xor bx,bx

xor cx,cx

mov ah, 3Fh ; Lectura de fichero

mov bx, [Handler] ; Le mando el handler del fichero

mov cx, 7 ; Numbytes x leer, 7 por el tamaño del username

mov dx, mBufferNameReg

int 21h ; interrupcion DOS

jc Salir\_Error

Comprobacion\_Usuario:

lea si, [mUsuario]

lea di, [mBufferNameReg]

repe cmpsb ; compara 2 str de si == di

jne NoLeerPassword

je LeerPassword

NoLeerPassword:

xor ah,ah

xor dx,dx

xor bx,bx

xor cx,cx

mov ah, 3Fh ; Lectura de fichero

mov bx, [Handler] ; Le mando el handler del fichero

mov cx, 7 ; Leo otros 7 para llegar al sig username

int 21h ; interrupcion DOS

jc Salir\_Error

jmp Avanzar

LeerPassword:

;Lectura de la "Coma"

xor ah,ah

xor dx,dx

xor bx,bx

xor cx,cx

mov ah, 3Fh ; Lectura de fichero

mov bx, [Handler] ; Le mando el handler del fichero

mov cx, 1 ; Me salto una coma

int 21h ; interrupcion DOS

jc Salir\_Error

;Lectura de Password

xor ah,ah

xor dx,dx

xor bx,bx

xor cx,cx

mov ah, 3Fh ; lectura de fichero

mov bx,[Handler] ; Le mando el handler del fichero

mov cx, 5 ; Leo la pass de 5 chars

mov dx, mPasswordR ; Le mando el buffer para guardar

int 21h ; interrupcion DOS

jc Salir\_Error

Comprobacion\_Password:

lea si, [mPasswordLogin]

lea di, [mPasswordR]

repe cmpsb ; compara 2 str de si == di

je Ingreso\_Correcto

jne No\_Existe\_Pass

Avanzar:

cmp ax, 0

jne EncontrarUsuario

je No\_Existe\_Usuario

No\_Existe\_Usuario:

println mMsgErrorUsuario

call Close\_File

call readline

jmp \_start

No\_Existe\_Pass:

println mMsgErrorPass

call Close\_File

call readline

jmp \_start

Ingreso\_Correcto:

call Close\_File

call cls

jmp start

ret

toppts:

call oArchReportes

mov di, 0 ; Inicializa cont = 0

call cls

println mHeader

println mEnter

println mTopPts

println mEnter

For:

inc di

xor ah,ah

xor dx,dx

xor bx,bx

xor cx,cx

mov ah, 3Fh ; Lectura de fichero

mov bx, [Handler] ; Le mando el handler del fichero

mov cx, 16 ; Numbytes x leer, 7 por el tamaño del username

mov dx, mBufferQChar

int 21h ; interrupcion DOS

jc Salir\_Error

println mBufferQChar

cmp di, 10 ; IF (bytes trasferidos){ NADA.. } else -> jne

jne For

fin:

call Close\_File

println mEnter

call readline

jmp \_start

ret

toptime:

call oArchReportes

mov di, 0 ; Inicializa cont = 0

call cls

println mHeader

println mEnter

println mTopTiempo

println mEnter

For2:

inc di

xor ah,ah

xor dx,dx

xor bx,bx

xor cx,cx

mov ah, 3Fh ; Lectura de fichero

mov bx, [Handler] ; Le mando el handler del fichero

mov cx, 16 ; Numbytes x leer, 7 por el tamaño del username

mov dx, mBufferQChar

int 21h ; interrupcion DOS

jc Salir\_Error

println mBufferQChar

cmp di, 10 ; IF (bytes trasferidos){ NADA.. } else -> jne

jne For2

fin2:

call Close\_File

println mEnter

call readline

jmp \_start

ret

comparar\_entrada:

cmp al,31h ; If(keypressed == 1) Then

je verificar\_usuario

cmp al,32h ; If(keypressed == 2) Then

je registrar\_usuario

cmp al,33h ; If(keypressed == 3) Then

je Salir

jmp \_start ; else{ ir a Start W}

ret

comparar\_entradaa:

cmp al,31h ; If(keypressed == 1) Then

je toppts

cmp al,32h ; If(keypressed == 2) Then

je toptime

cmp al,33h ; If(keypressed == 3) Then

je Salir

jmp \_start ; else{ ir a Start W}

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;; J U E G O D E L A S E R P I E N T E ;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; ========================================================================

start:

call AssignASCII

call clearscreen

mov di, 48

repeatit:

call PrintSnake

call KeyOperation

call FoodCheck

call CollisionCheck

call clearscreen

println mUserLogin

println punteo

jmp repeatit

exit:

mov ax,0x4c00

int 0x21

; ========================================================================

sumaruno:

cmp di, 57 ; Si es == 9

je resetToZero ; set = 0

jne incrementar ; contador++

resetToZero:

mov di, 48

inc di

mov [punteo+7], di

mov di, 47

incrementar:

inc di

mov [punteo+8], di ;asignar el siguiente dato del arreglo tamaño byte

ret

; Delay subroutine needs to be modifed according to dosbox and NASM clockcycles.....

delay:

push ax

push bx

mov bx,0

outerdelay:

mov ax,0

innerdelay:

add ax,1

cmp ax,100

jne innerdelay

add bx,1

cmp bx,400

jne outerdelay

pop bx

pop ax

ret

GetKey:

mov ah,1

int 0x16

jz leavethis

mov [newthing],ah

mov ax,0

int 0x16

jmp leave1

leavethis:

mov ah,[newthing]

leave1:

call delay

call delay

call delay

call delay

ret

terminate:

call clearscreen

mov ah,0x13

mov al,1

mov bh,0

mov bl,7

mov dx,0x0c08 ;; erminate ;;

mov cx,40

push cs

pop es

mov bp,mmsgfinjuego

int 0x10

call delay

call delay

call delay

jmp exit

ret

blink:

jmp keyagain

ret

CollisionCheck:

push bx

push ax

mov bh,0

mov bl,[strlen]

shl bx,1

sub bx,2

mov ax,[string+bx]

againCollisionCheck:

sub bx,2

cmp ax,[string+bx]

je terminate

cmp bx,0

je exitCollisionCheck

jmp againCollisionCheck

exitCollisionCheck:

pop ax

pop bx

ret

KeyOperation:

push bx

call GetKey

mov bh,0

mov bl,[strlen]

sub bl,1

shl bx,1 ; multiply by 2

mov dx,[string+bx]

cmp ah,0x48

jne NextKeyOperation2

cmp dl,0 ; give new head position

je exit

cmp byte[direction],3

je blink

sub dl,1

mov byte[newthing],0x48

mov [string+bx],dx

mov byte[direction],1

jmp exitKeyOperation

NextKeyOperation2:

cmp ah,0x4d

jne NextKeyOperation3

cmp dh,79 ; give new head position

je exit

cmp byte[direction],4

je blink

add dh,1

mov byte[newthing],0x4d

mov [string+bx],dx

mov byte[direction],2

jmp exitKeyOperation

NextKeyOperation3:

cmp ah,0x50

jne NextKeyOperation4

cmp dl,24 ; give new head position

je exit

cmp byte[direction],1

je blink

add dl,1

mov byte[newthing],0x50

mov [string+bx],dx

mov byte[direction],3

jmp exitKeyOperation

NextKeyOperation4:

cmp ah,0x4b

jne keyagain

cmp dh,0 ; give new head position

je exit

cmp byte[direction],2

je blink

sub dh,1

mov byte[newthing],0x4b

mov [string+bx],dx

mov byte[direction],4

jmp exitKeyOperation

keyagain:

call KeyOperation

exitKeyOperation:

pop bx

ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

AssignASCII:

; DESCRIPRION (first 4 asciis contain heads followed by 4 tails,

; vertical and horizonals straight bodies and then four curved bodies,

; -so total is 14)s

push ax

push bx

push cx

push dx

push ds ;

pop es ; make sure ES = DS

mov bp, OurFont ;

mov cx,15 ; we'll change just 14 of them (read the description of the sub-routine)

mov dx,1 ; STARTING ASCII NUMBER

mov bh,16 ; 16 bytes per char

xor bl,bl ; RAM block

mov ax,1100h ; change font to our font

int 10h ; video interrupt

push 0xb800

pop es

pop dx ;;; 2 times down also!!!!!!

pop cx

pop bx

pop ax

ret

mov byte[es:00],1

mov byte[es:4],2

mov byte[es:8],3

mov byte[es:12],4

mov byte[es:16],5

mov byte[es:20],6

mov byte[es:24],7

mov byte[es:28],8

mov byte[es:32],9

mov byte[es:36],10

mov byte[es:00],14

mov byte[es:02],11

mov byte[es:160],13

mov byte[es:162],12

pop dx

pop cx

pop bx

pop ax

ret ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

PrintSnake: ; calls body printing functions one by one ;;

;;

call PrintHead ;;

call PrintTrunk ;;

call PrintTail ;;

call ShiftLeft

call PrintFood

ret ;;

;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

PrintFood:

push bx

push di

mov bx,[FoodPtr]

mov di,[Food+bx]

mov byte[es:di],15

pop di

pop bx

ret

FoodCheck:

push ax

push bx

push cx

push dx

push si

push di

mov al,[strlen]

call Position

mov si,ax ; copy position of the head

mov bx,[FoodPtr]

mov di,[Food+bx]

cmp si,di

jne exitFoodCheck

mov bh,0

mov bl,[strlen]

shl bx,1

sub bx,2

againFoodloop:

mov ax,[string+bx] ; shiftig right loop to increase length

mov [string+bx+2],ax

sub bx,2

cmp bx,-2

jne againFoodloop

mov ax,[DiscardedTail]

mov [string],ax

add byte[strlen],1

add word[FoodPtr],2

cmp word[FoodPtr],12

jna soquela

mov word[FoodPtr],0

soquela:

pop di

pop si

pop dx

pop cx

pop bx

pop ax

call sumaruno

jmp retorna

exitFoodCheck:

pop di

pop si

pop dx

pop cx

pop bx

pop ax

retorna:

ret

PrintHead:

push ax

push bx

push 0xb800

pop es

mov al,[strlen] ; requirement of the function

mov [StrPosition],al

call Position ; this function returns postion in ax (so for safety ax is pushed in stack at the start!)

mov bx,ax ; copy position is pc

cmp byte[direction],0x1

jne nextcmp2

mov al,1 ; the ascii of head directed upwards

mov [es:bx],al

mov [SubDirection],al ; update 'subdirection' memory place

jmp headexit

nextcmp2:

cmp byte[direction],0x2

jne nextcmp3

mov al,2 ; the ascii of head directed rightwards

mov [es:bx],al

mov [SubDirection],al ; update 'subdirection' memory place

jmp headexit

nextcmp3:

cmp byte[direction],0x3

jne nextcmp4

mov al,3 ; the ascii of head directed downwards

mov [es:bx],al

mov [SubDirection],al ; update 'subdirection' memory place

jmp headexit

nextcmp4:

mov al,4 ; the ascii of head directed leftwards

mov [es:bx],al

mov [SubDirection],al ; update 'subdirection' memory place

headexit:

pop bx

pop ax

ret

PrintTrunk: ;

;Description (this prints the whole body of the snake) ;

againPrintTrunk:

cmp byte[StrPosition],2

je exitPrintTrunk

sub byte[StrPosition],1 ;

call TrunkOrganPrint ;

jmp againPrintTrunk ;

exitPrintTrunk:

ret

TrunkOrganPrint: ;

;Description (this function ptints organs relative to body position)

mov al,[StrPosition]

call Position

mov bx,ax ; copy first position position in bx

mov al,[StrPosition]

sub al,1

call Position

cmp byte[SubDirection],1

jne OrganCmp2 ; this is main comparison

sub ax,2

cmp ax,bx

jne sub1num2 ; this is sub comparison

mov al,13

mov [es:bx],al

mov byte[SubDirection],4

jmp exitTrunkOrganPrit

sub1num2:

sub ax,158

cmp ax,bx

jne sub1num3

mov al,9

mov [es:bx],al

jmp exitTrunkOrganPrit

sub1num3:

mov al,12

mov [es:bx],al

mov byte[SubDirection],2

jmp exitTrunkOrganPrit

OrganCmp2:

cmp byte[SubDirection],2

jne OrganCmp3 ; this is main comparison

add ax,160

cmp ax,bx

jne sub2num2 ; this is sub comparison

mov al,13

mov [es:bx],al

mov byte[SubDirection],3

jmp exitTrunkOrganPrit

sub2num2:

sub ax,158

cmp ax,bx

jne sub2num3

mov al,10

mov [es:bx],al

jmp exitTrunkOrganPrit

sub2num3:

mov al,14

mov [es:bx],al

mov byte[SubDirection],1

jmp exitTrunkOrganPrit

OrganCmp3:

cmp byte[SubDirection],3

jne OrganCmp4 ; this is main comparison

sub ax,2

cmp ax,bx

jne sub3num2 ; this is sub comparison

mov al,14

mov [es:bx],al

mov byte[SubDirection],4

jmp exitTrunkOrganPrit

sub3num2:

add ax,162

cmp ax,bx

jne sub3num3

mov al,9

mov [es:bx],al

jmp exitTrunkOrganPrit

sub3num3:

mov al,11

mov [es:bx],al

mov byte[SubDirection],2

jmp exitTrunkOrganPrit

OrganCmp4:

; SubDirection is obviously 4

add ax,160

cmp ax,bx

jne sub4num2 ; this is sub comparison

mov al,12

mov [es:bx],al

mov byte[SubDirection],3

jmp exitTrunkOrganPrit

sub4num2:

sub ax,162

cmp ax,bx

jne sub4num3

mov al,10

mov [es:bx],al

jmp exitTrunkOrganPrit

sub4num3:

mov al,11

mov [es:bx],al

mov byte[SubDirection],1

exitTrunkOrganPrit:

ret

PrintTail: ;

push ax

push bx ;

;Description (prints the tail of the snake according to 'SubDirection') ;

mov al,1

call Position ; function returns position in ax

mov bx,ax ; copy

cmp byte[SubDirection],1

jne NextTailCmp2

mov al,5

mov [es:bx],al

jmp exitPrintTail

NextTailCmp2:

cmp byte[SubDirection],2

jne NextTailCmp3

mov al,6

mov [es:bx],al

jmp exitPrintTail

NextTailCmp3:

cmp byte[SubDirection],3

jne NextTailCmp4

mov al,7

mov [es:bx],al

jmp exitPrintTail

NextTailCmp4:

mov al,8

mov [es:bx],al

exitPrintTail:

pop bx ;

pop ax ;

ret

Position:

push bx

mov bh,0

mov bl,al ; moves the length of the string in bx

mov ax,0 ; make sure that ah has 0 in it

shl bx,1 ; multiply by two as we are dealing with word

sub bx,2 ; start direction the head position

mov al,80

push bx

mov bl,[string+bx] ;load y-coordinate from the memore

mul bl

pop bx

mov bl,[string+bx+1] ;multiplication algo (leading x-coordinate)

mov bh,0 ; make sure that bh shas 0 for proper addition with ax

add ax,bx

shl ax,1

pop bx

ret

ShiftLeft: ;

push ax

push bx

push cx

mov bx,[string]

mov [DiscardedTail],bx

mov ch,0 ;make sure there is nothing else in ch

mov cl,[strlen]

mov bx,0

sub cx,1

ShiftLeftAgain:

mov ax,[string+bx+2]

mov [string+bx],ax

add bx,2

sub cx,1

cmp cx,0

jne ShiftLeftAgain

pop cx

pop bx

pop ax

ret

clearscreen:

push es

push di

mov di,0

push 0xb800

pop es

mov ax,0x0720

mov cx,2000

rep stosw

pop di

pop es

ret

; for my own sake I have made it not to word with any background attribute as horizontal characters are different from verticle ones!!!

; ascii code 1

OurFont db 10000001b

db 10000001b

db 10000001b

db 11000011b

db 11000011b

db 11100111b

db 10100111b

db 10011111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

; ascii code 2

db 00000000b

db 00000000b

db 11111111b

db 11110010b

db 11110100b

db 11111000b

db 11110000b

db 11100000b

db 11100000b

db 11110000b

db 11111000b

db 11111100b

db 11111110b

db 11111111b

db 00000000b

db 00000000b

; ascii code 3

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 10011111b

db 10100111b

db 11100111b

db 11000011b

db 11000011b

db 10000001b

db 10000001b

db 10000001b

; ascii code 4

db 00000000b

db 00000000b

db 11111111b

db 01110011b

db 00110111b

db 00011111b

db 00001111b

db 00000111b

db 00000111b

db 00001111b

db 00011111b

db 00111111b

db 01111111b

db 11111111b

db 00000000b

db 00000000b

;;;;;;;;;;;;;;;;;;;;;;;;

;tail

; ascii code 5

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 01111110b

db 01111110b

db 01111110b

db 00111100b

db 00111100b

db 00111100b

db 00011000b

db 00011000b

db 00011000b

; ascii code 6

db 00000000b

db 00000000b

db 00000000b

db 00000011b

db 00001111b

db 00011111b

db 01111111b

db 11111111b

db 11111111b

db 01111111b

db 00011111b

db 00001111b

db 00000011b

db 00000000b

db 00000000b

db 00000000b

; ascii code 7

db 00011000b

db 00011000b

db 00011000b

db 00111100b

db 00111100b

db 00111100b

db 01111110b

db 01111110b

db 01111110b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

; ascii code 8

db 00000000b

db 00000000b

db 00000000b

db 11000000b

db 11110000b

db 11111000b

db 11111110b

db 11111111b

db 11111111b

db 11111110b

db 11111000b

db 11110000b

db 11000000b

db 00000000b

db 00000000b

db 00000000b

;;;;;;;;;;;;;; vartical followed by horizontal

; ascii code 9

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111be

; ascii code 10

db 00000000b

db 00000000b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 00000000b

db 00000000b

;;;;;;;;;;;;;;;;;;;;;;; turns

;opens on left and dows

; ascii code 11

db 00000000b

db 00000000b

db 11110000b

db 11111000b

db 11111100b

db 11111110b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

; opens from up and left

; ascii code 12

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111110b

db 11111100b

db 11111000b

db 11110000b

db 00000000b

db 00000000b

;opens from right and top

; ascii code 13

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 0111111b

db 00111111b

db 00011111b

db 00001111b

db 00000000b

db 00000000b

; opens from bottom and right

; ascii code 14

db 00000000b

db 00000000b

db 00001111b

db 00011111b

db 00111111b

db 01111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

db 11111111b

;;;;;;;;;;;;;;

; Food (apple) ascii 15 decimal

db 00000000b

db 00000000b

db 01000000b

db 01100000b

db 00010000b

db 01101010b

db 11111111b

db 11111111b

db 11111111b

db 01111110b

db 01111110b

db 00011000b

db 00000000b

db 00000000b

db 00000000b

db 00000000b

Environment

DOSBox 0.74

NASM

Sistema Operativo

* Compilado
  + Windows 10.0.14393
  + Procesador i7-4700HQ @ 2.4 GHz
  + 16 GiB Ram