Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа № 4

по дисциплине ПнаЯА

Вариант 1

Выполнил студент гр. 250502: Бекетова М.А.

Проверил: Туровец Н.О.

Минск 2023

1 ТЕОРИТИЧЕСКИЕ СВЕДИНИЯ

В большинстве текстовых видеорежимов под видеопамять отводится специальная область памяти, начинающаяся с абсолютного адреса B800h:0000h и заканчивающаяся адресом B800h:FFFFh.

В текстовых режимах для хранения каждого изображенного символа используются два байта: -- байт с ASCII-кодом символа; -- байт атрибута символа (указывает цвет символа и фона, мигание).

Для установки требуемого программе видеорежима используется прерывание 10h (видеосервис) BIOS

Обработка нажатий на клавиатуру может производиться различными способами:

-- с помощью прерываний ввода символов DOS;

-- с помощью прерываний ввода символов BIOS;

-- с помощью прямого доступа к буферу клавиатуры

-- с помощью доступа к портам ввода-вывода клавиатуры.

Каждой клавише на клавиатуре соответствует уникальный код, называемый скан-код.

Персональный компьютер содержит два устройства для управления процессами:

-- часы реального времени (RTC)

-- системный таймер

2 КОД

.model small

.data

snake dw 0302h ;tail (03,02)

dw 0303h ;tail (03,03)

dw 0304h ;tail (03,04)

dw 0305h ;tail (03,05)

dw 0306h ;head (03,06)

dw 7CCh dup('?') ;7 dw elements

MOVING\_UP equ 0FF00h ;symbol const

MOVING\_DOWN equ 0100h

MOVING\_RIGHT equ 0001h

MOVING\_LEFT equ 0FFFFh

up\_boder\_sign db 0C4h ;symbol of up border

start\_pos dw 0

count dw 00h

count\_digit db 7 dup('0') ;array of 7 db elements

.stack 100h

.code

;;;;;;;;;;PROCEDURES;;;;;;;;;;

delay proc

push cx

mov ah, 0

int 1Ah ;to get the current system time

add dx, 3

mov bx, dx

repeat:

int 1Ah

cmp dx, bx

jl repeat

pop cx

ret

delay endp

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

key\_press proc

is\_key\_not\_pressed:

mov ax, 0100h ;check for keyboard buffer

int 16h ;responsible for keyboard input

jz end\_of\_key\_press ;if no key has been pressed

xor ah, ah

end\_game:

int 16h

cmp ah, 01h ;cmp to Esc

jne move\_down ;if not equal

jmp end\_of\_game ;end the game

move\_down:

cmp ah, 50h ;down key

jne move\_up ;go to next label

cmp cx, MOVING\_UP ;if move to itself

je end\_of\_key\_press

mov cx, MOVING\_DOWN ;move down

jmp end\_of\_key\_press

move\_up:

cmp ah, 48h ;up key

jne move\_left ;to next label

cmp cx, MOVING\_DOWN

je end\_of\_key\_press

mov cx, MOVING\_UP

jmp end\_of\_key\_press

move\_left:

cmp ah, 4Bh ;left key

jne move\_right

cmp cx, MOVING\_RIGHT;if move to itself

je end\_of\_key\_press

mov cx, MOVING\_LEFT

jmp end\_of\_key\_press

move\_right:

cmp cx, MOVING\_LEFT

je end\_of\_key\_press

mov cx, MOVING\_RIGHT

end\_of\_key\_press:

ret

key\_press endp

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

spawn\_food proc

push cx ;to save value

mov ah, 0

int 1Ah ;to get current time (stores in dl)

mov bl, dl ;move this time to bl

mov cx, 01h ;enter a loop at least once

check\_for\_random\_number:

inc bl

cmp bx, 1

jg continue\_check

inc bx

jmp check\_for\_random\_number

continue\_check:

cmp bx, 78

jng write\_coordinates

shr bl, 1 ;divide by 2

jmp check\_for\_random\_number

write\_coordinates:

mov dl, bl ;X

check\_for\_random\_number2: ;generating number for Y

cmp bx, 13h

jg div\_bx

cmp bx, 3h

jl inc\_bx

jmp write\_coordinates2

div\_bx:

shr bl, 2 ;move number by 2 places to right

jmp check\_for\_random\_number2

inc\_bx:

inc bl

jmp check\_for\_random\_number2

write\_coordinates2: ;Y

mov dh, bl

mov ax, 0200h ;cursor position

int 10h

mov ax, 0800h ;display

int 10h

cmp al, 2Ah ;cmp \* and al

je check\_for\_random\_number ;if equal generate XY again

mov ax, 090Ch

mov bl, 0Ch ;red color

int 10h

pop cx ;return orogonal cx

ret

spawn\_food endp

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

chech\_for\_game\_over proc

cmp al, 0C4h

je move\_around\_vertical

cmp al, 0DBh

je move\_around\_horizontal

cmp al, 02Ah ;cmp to \*

je game\_over

jmp continue

move\_around\_vertical:

cmp dh, 01h ;row == 1

je upper\_border

lower\_borde:

mov dh, 02h ;go from upper border

jmp move\_cursor

upper\_border: ;go from lower border

mov dh, 20

jmp move\_cursor

move\_around\_horizontal:

cmp dl, 00h ;column == 0

je left\_border

right\_border: ;go from left border

mov dl, 01h

jmp move\_cursor

left\_border: ;go from right border

mov dl, 78

jmp move\_cursor

move\_cursor:

push ax

mov ah, 02h

int 10h

pop ax

jmp continue

game\_over:

mov ax, 4c00h ;function to terminate the program

int 21h ;and to exit

continue:

mov [snake+si], dx

ret

chech\_for\_game\_over endp

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

itoa proc

pusha

mov di, offset count\_digit

add di, 5

xor bx, bx

mov ax, word ptr [count]

outer\_loop:

mov bx, 10

xor dx, dx

div bx

add dl, '0'

mov [di], dl

dec di

cmp ax, 0

je end\_outer\_loop

jmp outer\_loop

end\_outer\_loop:

popa

ret

itoa endp

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

set\_count proc

pusha ;all regs to stack

call itoa

mov bx, 148 ;position

mov cx, 6

;mov di, offset count\_digit

xor di, di

outp\_count:

mov al, byte ptr[count\_digit + di] ;pointer to digit

mov byte ptr es:[bx], al ;put digit

inc di

add bx, 2

loop outp\_count

popa

ret

set\_count endp

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

increment\_count proc ;increments score

pusha

mov ax, word ptr[count]

inc ax

mov word ptr[count], ax

call itoa

call set\_count

popa

ret

increment\_count endp

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

paint\_game\_area proc

mov ah, 02h ;position of cursor

mov dh, 01h ;row

mov dl, 0h ;column

int 10h

paint\_upper\_border:

mov ax, 09C4h ;symbol

mov bl, 02h ;color

mov cx, 80

int 10h

paint\_left\_border:

mov ah, 02h

mov dh, 20 ;row

mov dl, 00h

int 10h

mov cx, 20

paint\_left\_border\_loop:

push cx

mov ax, 09DBh ;symbol

mov bl, 02h ;color

mov cx, 1

int 10h

pop cx

mov ah, 02h

mov dh, cl ;down 1 row

mov dl, 00h

int 10h

loop paint\_left\_border\_loop

paint\_right\_border:

mov ah, 02h

mov dh, 20 ;row

mov dl, 79 ;column

int 10h

mov cx, 20

paint\_right\_border\_loop:

push cx

mov ax, 09DBh ;synbol

mov bl, 02h ;color

mov cx, 1

int 10h

pop cx

mov ah, 02h

mov dh, cl

mov dl, 79

int 10h

loop paint\_right\_border\_loop

paint\_lower\_border:

mov ah, 02h

mov dh, 21

mov dl, 0

int 10h

mov ax, 09C4h

mov bl, 02h

mov cx, 80

int 10h

ret

paint\_game\_area endp

;;;;;;;;;;;;;;;END;;;;;;;;;;;;;;

start:

mov ax, @data

mov ds, ax

mov ax, 0B800h ;load the value 0B800H

mov es, ax ;accessing the video memory, where the game area will be displayed

mov ax, 0003h ;sets video mode to 80x25 text mode with 16 colors

int 10h ;int 10h interrupt to set the desired video mode

call paint\_game\_area ;for painting the game area

call set\_count ;to initialize and display the count variable

mov ah, 02h

mov dx, 0302h ;pointer position (3,2)

int 10h

mov cx, 5 ;number of characters to display

mov ah, 09h ;

mov bl, 0Eh ;color [0ah - green]

mov al, 002Ah ;\*

int 10h

mov si, 8 ;head coordinates

xor di, di

mov cx, MOVING\_RIGHT ;cx stores direction

call spawn\_food

game\_cycle:

call delay

call key\_press ;delay between iterations

xor bh, bh

mov ax, word ptr [snake+si] ;current head position

add ax, cx ;to calculate new head position

add si, 2 ;+2(dw) to move to the next pos in the snake array

mov dx, ax ;new head position

mov ax, 0200h ;move pointer to new head position

int 10h

mov ax, 0800h ;read symbol in head (character at pointer pos)

int 10h

call chech\_for\_game\_over ;check if the game should end

mov dh, al ;res is stored in dh

mov ah, 09h

mov bl, 0Eh ;color

push cx

mov al, 002Ah ;\*

mov cx, 1 ;single char will be displayed

int 10h

pop cx

check\_for\_eating\_food:

cmp dh, 0Ch ;head symbol and dh register

jne remove\_tail ;if not equal, snake has not eaten food

call increment\_count ;count++

call spawn\_food ;new food

jmp game\_cycle ;game loop

remove\_tail:

mov ax, 0200h ;move pointer to new position

mov dx, [snake+di] ;to this position i mean

int 10h

mov ax, 0200h ;move pointer to new position

mov dl, 0020h ;print spase (to cut tail)

int 21h

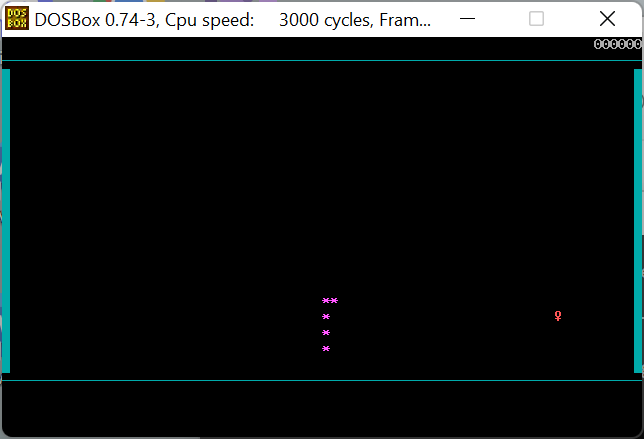
add di, 2

jmp game\_cycle

end\_of\_game:

end start

3 ДЕМОНСТРАЦИЯ РАБОТЫ



4 ВЫВОДЫ

Таким образом была реализована программа игра «Змейка». В результате удалось ознакомиться в рамках создания видеоигры с обработкой нажатий кнопок клавиатуры, рассмотреть прямой доступ к видеопамяти с целью формирования игрового поля и информации для пользователя.