

Лабораторная работа №13.

Программирование игр. Включение файлов

Макаров П. А.

23 ноября 2021 г.

1 Задания для самостоятельного решения

1. Напишите следующую программу, ассемблируйте её с помощью `nasm`. Выясните смысл каждой написанной строки. Исследуйте работу получившейся программы. Как улучшить читаемость кода?

Файл `main.asm`

```
org 0x100

start:    call initscr
          call play
          call initscr
          cmp byte [player], 1
          je TiedG
          cmp byte [player], 0
          je P11Win
          jmp P12Win
exit:     mov ah, 0
          int 0x16
          call exitscr
          int 0x20

P11Win:   mov si, MSG1
          mov bl, 4
          jmp NextChar

P12Win:   mov si, MSG2
          mov bl, 14
```

```

        jmp NextChar

TiedG:  mov si, MSG3
        jmp NextChar

NextChar: mov al, byte [si]
        cmp al, 0
        je exit
        mov ah, 0x09
        mov bh, 0
        mov cx, 1
        int 0x10
        inc si
        inc dl
        mov ah, 02
        int 0x10
        jmp NextChar

```

```

MSG1: db "Player 1 wins!", 0
MSG2: db "Player 2 wins!", 0
MSG3: db "Tied game.", 0

```

```

#include "screen.asm"
#include "play.asm"

```

Файл screen.asm

```

initscr: mov ax, 0x0003      ; set color text 80x25 video mode
        int 0x10

        mov ax, 0x0920      ; set attribute to all screen
        mov bx, 0x0002
        mov cx, 2000
        int 0x10

        mov ax, 0x0EC9      ; left top symbol
        mov bh, 0x00
        int 0x10

        mov ah, 02
        mov bh, 0
        mov dx, 0x0001

```

```

int 0x10

mov ax, 0x0ACD      ; top line
mov bh, 0x00
mov cx, 78
int 0x10

mov ah, 02
mov bh, 0
mov dx, 0x004E
int 0x10

mov ax, 0x0EBB      ; right top symbol
mov bh, 0x00
int 0x10

mov ah, 02
mov bh, 0
mov dx, 0x1800
int 0x10

mov ax, 0x0EC8      ; left bottom symbol
mov bh, 0x00
int 0x10

mov ax, 0x0ACD      ; top line
mov bh, 0x00
mov cx, 78
int 0x10

mov ah, 02
mov bh, 0
mov dx, 0x184E
int 0x10

mov ax, 0x0EBC      ; right bottom symbol
mov bh, 0x00
int 0x10

leftcol_s: mov cx, 23
           mov dx, 0x0100

```

```

leftcol_n:  push cx
            mov ah, 02
            mov bh, 0
            int 0x10
            mov ax, 0x0ABA
            mov bh, 0x00
            mov cx, 1
            int 0x10
            inc dh
            pop cx
            loop leftcol_n

rightcol_s: mov cx, 23
            mov dx, 0x014E
rightcol_n: push cx
            mov ah, 02
            mov bh, 0
            int 0x10
            mov ax, 0x0ABA
            mov bh, 0x00
            mov cx, 1
            int 0x10
            inc dh
            pop cx
            loop rightcol_n

            mov ah, 01
            mov cx, 0x1F01
            int 0x10

center:    mov ah, 02
            mov dx, 0x0B28
            mov bh, 0
            int 0x10

            ret

exitscr:   mov ax, 0x0003      ; set color text 80x25 video mode
            int 0x10

            mov ah, 01

```

```
    mov cx, 0x1F1F
    int 0x10

    ret
```

Файл play.asm

```
play:    xor ah, ah
         int 0x16
         cmp al, 'q'
         je TiedQuit
         cmp ah, UPARR
         je up
         cmp ah, DOWNARR
         je down
         cmp ah, LEFTARR
         je left
         cmp ah, RIGHTARR
         je right
         cmp al, ENTER
         je isspace

         jmp play

TiedQuit: mov byte [player], 1
quit:    ret

up:      mov ah, 03
         mov bh, 0
         int 0x10
         cmp dh, 01
         je play
         dec dh
         mov ah, 02
         int 0x10
         jmp play

down:    mov ah, 03
         mov bh, 0
         int 0x10
         cmp dh, 23
         je play
```

```

        inc dh
        mov ah, 02
        int 0x10
        jmp play

left:   mov ah, 03
        mov bh, 0
        int 0x10
        cmp dl, 01
        je play
        dec dl
        mov ah, 02
        int 0x10
        jmp play

right:  mov ah, 03
        mov bh, 0
        int 0x10
        cmp dl, 78
        je play
        inc dl
        mov ah, 02
        int 0x10
        jmp play

isspace: mov ah, 08
        mov bh, 0
        int 0x10
        cmp al, SPACE
        jne play

        mov ah, 09
        mov bh, 0
        mov cx, 1
        cmp byte [player], 0
        jne player2
player1: mov al, '+'
        mov bl, 4
        jmp print
player2: mov al, 'o'
        mov bl, 14

```

```

print:    int 0x10
          jmp check

check:    mov ah, 03
          mov bh, 0
          int 0x10
          mov [x], dl
          mov [y], dh
          mov byte [S], '+'
          cmp byte [player], 0
          je ScanUp
          mov byte [S], 'o'
ScanUp:   mov ah, 02
          dec dh
          int 0x10
          mov ah, 08
          mov bh,
          int 0x10
          cmp al, [S]
          jne ScanRight
          inc byte [N]
          cmp byte [N], 5
          jl ScanUp
          ret
ScanRight: mov ah, 03
          mov bh,
          int 0x10
          mov [xtop], dh
          mov byte [N], 1
          mov dl, [x]
          mov dh, [y]
ScanRight1: mov ah, 02
          inc dl
          int 0x10
          mov ah, 08
          mov bh, 0
          int 0x10
          cmp al, [S]
          jne ScanDown
          inc byte [N]
          cmp byte [N], 5

```

```

        jl ScanRight1
        ret
ScanDown:  mov ah, 03
          mov bh, 0
          int 0x10
          mov [yright], dl
          mov byte [N], 1
          mov dl, [x]
          mov dh, [y]
ScanDown1: mov ah, 02
          inc dh
          int 0x10
          mov ah, 08
          mov bh, 0
          int 0x10
          cmp al, [S]
          jne ScanLeft
          inc byte [N]
          cmp byte [N], 5
          jl ScanDown1
          ret
ScanLeft:  mov ah, 03
          mov bh, 0
          int 0x10
          mov [xbottom], dh
          mov byte [N], 1
          mov dl, [x]
          mov dh, [y]
ScanLeft1: mov ah, 02
          dec dl
          int 0x10
          mov ah, 08
          mov bh, 0
          int 0x10
          cmp al, [S]
          jne CHECKP1
          inc byte [N]
          cmp byte [N], 5
          jl ScanLeft1
          ret
CHECKP1:  mov ah, 03

```



```

        mov bh, 0
        int 0x10
        mov [yleft], dl
        mov ah, 02
        mov dl, [x]
        mov dh, [y]
        int 0x10
        mov al, [xbottom]
        sub al, [xtop]
        cmp al, 6
        jge quit
        mov al, [yright]
        sub al, [yleft]
        cmp al, 6
        jge quit

        not byte [player]
        jmp play

player:  db 0
x:      db 0
y:      db 0
N:      db 1
S:      db '+'
xtop:   db 0
xbottom: db 0
yright: db 0
yleft:  db 0

SPACE:  equ 0x20
UPARR:  equ 0x48
DOWNARR: equ 0x50
RIGHTARR: equ 0x4D
LEFTARR: equ 0x4B
ENTER:  equ 0x0D

```

2. Улучшите написанную игру:

- разработайте процедуру, выводящую на начальном экране правила игры;
- задайте изменение цвета рамки экрана по ходу игры так, чтобы он совпадал с цветом текущего активного игрока;

- модернизируйте алгоритм программы для учёта линий не только по горизонтали и вертикали, но и по диагонали.

3. Напишите игру «Жизнь».