МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

ЛАБОРАТОРНАЯ РАБОТА №8

ОТЧЁТ

студента 2 курса 251 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Григорьева Данилы Евгеньевича

Проверено:

Старший преподаватель _____ Е. М. Черноусова

Саратов 2024

СОДЕРЖАНИЕ

1 Текст программы на языке ассемблера с комментариями

```
extrn GetStdHandle:proc, WriteConsoleW:proc, ReadConsoleW:proc,
lstrlenW:proc, wsprintfW:proc, ExitProcess:proc
STD_OUTPUT_HANDLE=-11
STD_INPUT_HANDLE=-10
wMAX=32767
wMIN=-32768
BUF_SIZE=1024
.data
    ; a =
    strA      db 61h, 00h, 20h, 00h, 3dh, 00h, 20h, 00h, 00h, 00h
    ; b =
    strB      db 62h, 00h, 20h, 00h, 3dh, 00h, 20h, 00h, 00h, 00h
    ; 0x5D + %d - %d = %d (prefix)
    strResult db 30h, 00h, 78h, 00h, 35h, 00h, 44h, 00h, 20h, 00h,
2bh, 00h, 20h, 00h, 25h, 00h, 64h, 00h, 20h, 00h, 2dh, 00h, 20h,
00h, 25h, 00h, 64h, 00h, 20h, 00h
              db 3dh, 00h, 20h, 00h
    strNum    db 25h, 00h, 64h, 00h, 0ah, 00h, 00h, 00h

    strLowest db 1ch, 04h, 38h, 04h, 3dh, 04h, 38h, 04h, 3ch, 04h,
30h, 04h, 3bh, 04h, 4ch, 04h, 3dh, 04h, 3eh, 04h, 35h, 04h, 20h,
00h, 47h, 04h, 38h, 04h, 41h, 04h
              db 3bh, 04h, 3eh, 04h, 3ah, 00h, 20h, 00h, 00h, 00h

    ; Ошибка ввода числа
    erStoi    db 1eh, 04h, 48h, 04h, 38h, 04h, 31h, 04h, 3ah, 04h,
30h, 04h, 20h, 00h, 32h, 04h, 32h, 04h, 3eh, 04h, 34h, 04h, 30h,
04h, 20h, 00h, 47h, 04h, 38h, 04h
              db 41h, 04h, 3bh, 04h, 30h, 04h, 0ah, 00h
              db 00h, 00h
    ; Введённый текст не является целым числом
    erNaN     db 12h, 04h, 32h, 04h, 35h, 04h, 34h, 04h, 51h, 04h,
3dh, 04h, 3dh, 04h, 4bh, 04h, 39h, 04h, 20h, 00h, 42h, 04h, 35h,
04h, 3ah, 04h, 41h, 04h, 42h, 04h
              db 20h, 00h, 3dh, 04h, 35h, 04h, 20h, 00h, 4fh, 04h,
32h, 04h, 3bh, 04h, 4fh, 04h, 35h, 04h, 42h, 04h, 41h, 04h, 4fh,
04h, 20h, 00h, 46h, 04h, 35h, 04h
              db 3bh, 04h, 4bh, 04h, 3ch, 04h, 20h, 00h, 47h, 04h,
38h, 04h, 41h, 04h, 3bh, 04h, 3eh, 04h, 3ch, 04h, 0ah, 00h
              db 00h, 00h
    ; Число слишком велико по модулю для дальнейших вычислений
    erOver    db 27h, 04h, 38h, 04h, 41h, 04h, 3bh, 04h, 3eh, 04h,
20h, 00h, 41h, 04h, 3bh, 04h, 38h, 04h, 48h, 04h, 3ah, 04h, 3eh,
```

```
04h, 3ch, 04h, 20h, 00h, 32h, 04h
                db 35h, 04h, 3bh, 04h, 38h, 04h, 3ah, 04h, 3eh, 04h,
20h, 00h, 3fh, 04h, 3eh, 04h, 20h, 00h, 3ch, 04h, 3eh, 04h, 34h,
04h, 43h, 04h, 3bh, 04h, 4eh, 04h
                db 20h, 00h, 34h, 04h, 3bh, 04h, 4fh, 04h, 20h, 00h,
34h, 04h, 30h, 04h, 3bh, 04h, 4ch, 04h, 3dh, 04h, 35h, 04h, 39h,
04h, 48h, 04h, 38h, 04h, 45h, 04h
                db 20h, 00h, 32h, 04h, 4bh, 04h, 47h, 04h, 38h, 04h,
41h, 04h, 3bh, 04h, 35h, 04h, 3dh, 04h, 38h, 04h, 39h, 04h, 0ah,
00h
                db 00h, 00h

    varA        dw ?
    varB        dw ?
    varResult dq ?
    hCin        dq ?
    hCout       dq ?
    buffer      dw BUF_SIZE / 2 dup (?)
    bufLen      dq 0
    number      dq ?
; 5Dh + A - B
.code
    initHandlers proc
        push rax
        push rcx
        sub rsp, 8
        mov rcx, STD_INPUT_HANDLE
        call GetStdHandle
        mov [hCin], rax
        mov rcx, STD_OUTPUT_HANDLE
        call GetStdHandle
        mov [hCout], rax
        add rsp, 8
        pop rcx
        pop rax
        ret
    initHandlers endp
    flush proc
        push rax
        push rcx
        lea rax, buffer
        mov rcx, BUF_SIZE
        clear_buffer:
            mov [rax + rcx], dl
```

```
        loop clear_buffer
    pop rcx
    pop rax
    ret
flush endp
cout proc
    push rax
    push rcx
    push rdx
    push r8
    push r9
    sub rsp, 28h
    mov rdx, rcx
    call lstrlenW
    mov r8, rax
    mov rcx, hCout
    xor r9, r9
    call WriteConsoleW
    add rsp, 28h
    pop r9
    pop r8
    pop rdx
    pop rcx
    pop rax
    ret
cout endp
cin proc
    push rcx
    push rdx
    push r8
    push r9
    sub rsp, 28h
    lea rax, buffer
    xor dl, dl
    call flush
    mov rcx, hCin
    lea rdx, buffer
    mov r8, BUF_SIZE
    lea r9, bufLen
    call ReadConsoleW
    mov rax, bufLen
    sub rax, 2
    mov bufLen, rax
    add rsp, 28h
```

```
    pop r9
    pop r8
    pop rdx
    pop rcx
    ret
cin endp

stoi proc
local num:word, sign:byte, i:byte, digit:word
    push rbx
    push rcx
    push rdx
    push r8
    push r9
    push rbp
    mov num,0
    mov sign,1
    mov i,0
    skip_leading_space:
        mov rax,bufLen
        cmp i,al
    jae parse_nan_error
        lea rax,buffer
        movzx rcx,i
        mov ax, word ptr [rax+rcx*2]
        mov bx, 20h
        cmp ax, bx
    jne check_for_sign
        inc cl
        mov i,cl
    jmp skip_leading_space
    check_for_sign:
        movsxd rax,dword ptr [bufLen]
        cmp i,al
        jae parse_nan_error
            lea rax, buffer
            movzx rcx,i
            mov dx,pword tr [rax+rcx*2]
            mov bx,2Dh
            cmp dx,bx
            je change_sign
                mov bx,2Bh
                cmp dx,bx
                jne convert_to_int
```

```
        change_sign:
            lea rax, buffer
            movzx rcx,i
            mov dx,word ptr [rax+rcx*2]
            mov bx,2Dh
            cmp dx,bx
            jne its_plus
                mov dl,-1
                jmp its_minus
            its_plus:
                mov dl,1
            its_minus:
            mov sign,dl
            mov al,i
            inc al
            mov i,al
convert_to_int:
    movsxd rax,dword ptr [bufLen]
    cmp i,al
    jae apply_sign_and_return
        lea rax,buffer
        movzx rcx,i
        mov ax,word ptr [rax+rcx*2]
        test ax,ax
    je apply_sign_and_return
    lea rax,buffer
    movzx rcx,i
    mov ax,word ptr [rax+rcx*2]
    mov bx, 20h
    cmp ax,bx
    jne check_is_digit
        mov cl, i
        inc cl
        mov i, cl
        jmp convert_to_int
    check_is_digit:
    lea rax,buffer
    movzx rcx,i
    mov ax,word ptr [rax+rcx*2]
    mov dx, 30h
    cmp ax, dx
    jl parse_nan_error
    mov dx, 39h
    cmp ax, dx
```

```
    jg parse_nan_error
    char_to_dword:
    lea rax,buffer
    movzx rcx,i
    mov ax,word ptr [rax+rcx*2]
    sub ax,30h
    mov digit,ax
    mov bx, wMAX / 10
    cmp num,bx
    jg parse_nan_error
    cmp num,bx
    jne build_number
        mov bx, 7
        cmp digit,bx
        jg parse_overflow_error
    build_number:
    mov ax, num
    mov bx, 0Ah
    imul bx
    mov bx, digit
    add ax, bx
    mov num,ax
    mov al,i
    inc al
    mov i,al
    jmp convert_to_int
apply_sign_and_return:
    mov ax, num
    movsx bx, sign
    imul ax, bx
    pop r9
    pop r8
    pop rdx
    pop rcx
    pop rbx
    ret
parse_nan_error:
    lea rcx, erNaN
    jmp parse_error
parse_overflow_error:
    lea rcx, erOver
parse_error:
    call cout
    xor rcx, rcx
```

```
        call ExitProcess
        ret
    ret
stoi endp

input proc
    push rax
    push rcx
    sub rsp, 8
    lea rcx, strA
    call cout
    call cin
    call stoi
    mov varA, ax

    lea rcx, strB
    call cout
    call cin
    call stoi
    mov varB, ax
    add rsp, 8
    pop rax
    pop rcx
    ret
input endp
calc proc
    push rax
    push rbx
    push rcx
    sub rsp, 8
    mov rax, 5Dh
    movsx rbx, varA
    movsx rcx, varB
    add rax, rbx
    sub rax, rcx
    mov varResult, rax
    add rsp, 8
    pop rcx
    pop rbx
    pop rax
    ret
calc endp
output proc
    push rax
```

```
push rbx
push rcx
push rdx
push r8
push r9
sub rsp, 30h
mov rbx, varResult
mov [rsp + 20h], rbx
lea rcx, buffer
lea rdx, strResult
movsx r8, varA
movsx r9, varB
call wsprintfW

lea rcx, buffer
call cout

lea rcx, strLowest
call cout

call flush
xor rdx, rdx
mov dx, varA
mov r8w, varB
cmp dx, r8w
jg b_is_lower
    lea rcx, strA
    call cout
    movsx r8, varA
    jmp cout_lowerest
b_is_lower:
    lea rcx, strB
    call cout
    movsx r8, varB
cout_lowerest:
lea rcx, buffer
lea rdx, strNum
call wsprintfW
lea rcx, buffer
call cout
add rsp, 30h
pop r9
pop r8
pop rdx
```

```
        pop rcx
        pop rbx
        pop rax
        ret
    output endp
    mainCRTStartup proc
        push rbp
        call initHandlers

        call input
        call calc
        call output

        xor rcx, rcx
        call ExitProcess
        pop rbp
        ret
    mainCRTStartup endp
end
```

2 Скриншот запуска программы