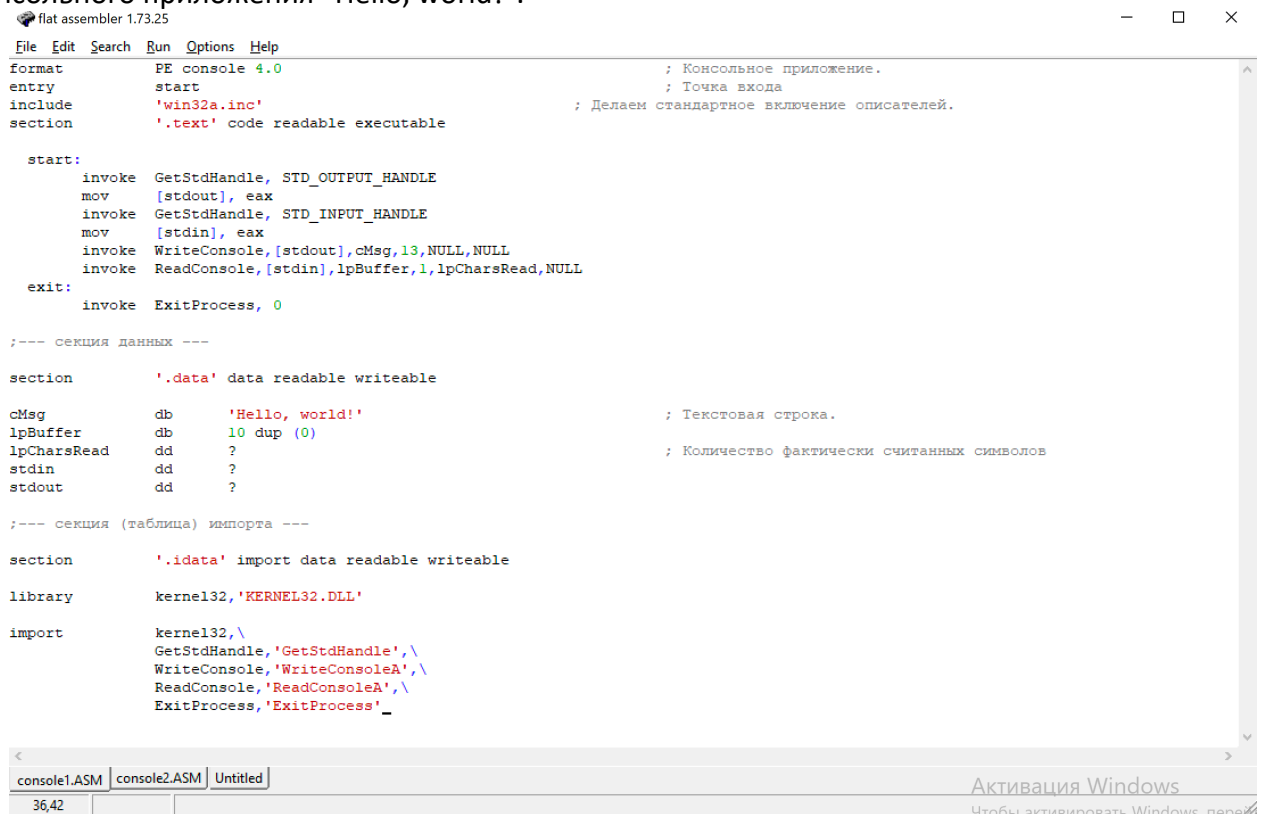


Домашняя работа 1.

1. Так как данный курс – мое первое знакомство с ассемблером, я решила начать с компиляции консольного приложения “Hello, world!”.



```
flat assembler 1.73.25
File Edit Search Run Options Help
format PE console 4.0 ; Консольное приложение.
entry start ; Точка входа
include 'win32a.inc' ; Делаем стандартное включение описателей.
section '.text' code readable executable

start:
    invoke GetStdHandle, STD_OUTPUT_HANDLE
    mov [stdout], eax
    invoke GetStdHandle, STD_INPUT_HANDLE
    mov [stdin], eax
    invoke WriteConsole, [stdout], cMsg, 13, NULL, NULL
    invoke ReadConsole, [stdin], lpBuffer, 1, lpCharsRead, NULL
exit:
    invoke ExitProcess, 0

;--- секция данных ---
section '.data' data readable writeable


cMsg db 'Hello, world!' ; Текстовая строка.
lpBuffer db 10 dup (0)
lpCharsRead dd ? ; Количество фактически считанных символов
stdin dd ?
stdout dd ?

;--- секция (таблица) импорта ---
section '.idata' import data readable writeable

library kernel32, 'KERNEL32.DLL'

import kernel32, \
    GetStdHandle, 'GetStdHandle', \
    WriteConsole, 'WriteConsoleA', \
    ReadConsole, 'ReadConsoleA', \
    ExitProcess, 'ExitProcess' _
```

Данная программа осуществляет вывод на консоли текста “Hello, world!”.

 C:\Users\astaskina\Documents\console1.EXE



Для закрытия консольного окна нужно нажать на любую клавишу.

2. Далее одной из самых распространенных задач для новичков в программировании является написание калькулятора. Ниже код калькулятора, в котором есть функции: сложения, вычитания,

умножения, деления, деления с остатком.

flat assembler 1.73.25

File Edit Search Run Options Help

```
format PE console 4.0
entry start
include 'win32a.inc'
section '.data' data readable writable

    msg1 db "Test %d",0dh,0ah,0

    entr db 'Enter %s ',0
    _1st db 'A:',0
    _2nd db 'B:',0
    _3rd db 'operation:',0
    infinity db 'infinity (well, this is a moot point...)', 0
    res db 'result: %d',0
    res2 db 'integer part %d',0
    nullsimbol db ' ',0
    res3 db '/%d',0
    itpt db ' %d',0
    nthn db '%d',0
    point db ', ',0
    A dd ?
    B dd ?
    C dd ?
```

section '.text' code readable executable

start:

```
cinvoke printf, entr, _1st
cinvoke scanf, itpt, A
cinvoke printf, entr, _2nd
cinvoke scanf, itpt, B
cinvoke printf, entr, _3rd
```

mov eax, '1'

```
invoke getch
cmp eax,43 ; '+' sign code
jnz @notAdd
    mov ecx, [A]
    add ecx, [B]
    cinvoke printf, res, ecx
    jmp @finish
```

@notAdd:

```
cmp eax,45 ; '-' sign code
jnz @notSub
```

console1.ASM console2.ASM

55,17

Активация Windows
Чтобы активировать Windows, перейдите в "Параметры".

flat assembler 1.73.25

File Edit Search Run Options Help

```
cmp eax,45 ; '-' sign code
jnz @notSub
    mov ecx, [A]
    sub ecx, [B]
    cinvoke printf, res, ecx
    jmp @finish
```

@notSub:

```
cmp eax,42 ; '*' sign code
jnz @notMul
    mov ecx, [A]
    imul ecx, [B]
    cinvoke printf, res, ecx
    jmp @finish
```

@notMul:

```
cmp eax,37 ; '/' sign code
jnz @notDiv
; Gives the answer as an integer quotient and a remainder,
; if remainder not equal to zero
    mov eax, [A]
    mov ecx, [B]
    mov edx, 0

    cmp [B], 0
    jnz @fractionNotZero
        cinvoke printf, infinity
        jmp @finish
    @fractionNotZero:

    div ecx
    mov [C], edx
    cinvoke printf, res, eax
    cinvoke printf, itpt, [C]
    cinvoke printf, res3, [B]
    jmp @finish
```

@notDiv:

```
cmp eax,47 ; '%' code sign
jnz @notDiv2
; Gives answer as a decimal fraction with comma as decimal separator
; and exactly 4 digits after separator
    mov eax, [A]
    mov ecx, [B]
    mov edx, 0
```

console1.ASM console2.ASM

55,17

Активация Windows
Чтобы активировать Windows, перейдите в "Параметры".

```
flat assembler 1.73.25
File Edit Search Run Options Help

mov eax, [A]
mov ecx, [B]
mov edx, 0

cmp [B], 0
jnz @fractionNotZero2
    cinvoke printf, infinity
    jmp @finish
@fractionNotZero2:

div ecx
mov [C], edx
cinvoke printf, res, eax
cinvoke printf, point ;prints decimal separator

mov ebx, 0
@loop:
    mov eax, [C]
    mov ecx, [B]
    imul eax, 10
    mov edx, 0
    div ecx
    mov [C], edx
    cinvoke printf, nthn, eax
    add ebx, 1

cmp ebx, 4 ;number of digits in fractional part
jnz @loop

jmp @finish
@notDiv2:


@finish:
invoke getch ;waiting for any key pressed
invoke ExitProcess, 0

section '.idata' import data readable
library kernel, 'kernel32.dll', \
msvcrt, 'msvcrt.dll'

import kernel, \
ExitProcess, 'ExitProcess'


import msvcrt, \
printf, 'printf', \
scanf, 'scanf', \
getch, '_getch'
```

При запуске приложение запрашивает параметр A и параметр B, а затем запрашивает операцию, которую нужно выполнить с этими двумя параметрами.

 C:\Users\astaskina\Documents\FASM\console2.EXE




Сложение 5 и 3:

 C:\Users\astaskina\Documents\FASM\console2.EXE

```
Enter A: 5
Enter B: 3
Enter operation: result: 8
```

Умножение 11 и 4:

 C:\Users\astaskina\Documents\FASM\console2.EXE

```
Enter A: 11
Enter B: 4
Enter operation: result: 44
```

Данное окно также закрывается по нажатию на любую клавишу.

3. Следующий шаг – программа, которая запрашивает данные на ввод с клавиатуры. Данное консольное приложение запрашивает ваше имя, а затем приветствует вас. Оно также закрывается по нажатию на любую клавишу.

```
flat assembler 1.73.25
File Edit Search Run Options Help
format PE console 4.0
entry main
include 'win32a.inc'
section '.data' data readable writeable


msg db "Enter your name: ",0
chrarr db "%s",0
msg2 db "Hello, %s",0dh,0ah,0
p db "pause",0

section '.code' code readable executable

main:
push ebp
mov ebp,esp
sub esp,24
mov dword [esp],msg
call [printf]
lea eax,[ebp-12]
mov dword [esp+4],eax
mov dword [esp],chrarr
call [scanf]
mov dword [esp],msg2
call [printf]
mov dword [esp],p
call [system]
mov dword [esp],0
call [exit]


section '.idata' import data readable
library msvcrt,'msvcrt.dll'

import msvcrt,\
printf,'printf',\
scanf,'scanf',\
system,'system',\
exit,'exit'
```

 C:\Users\astaskina\Documents\FASM\console3.EXE

```
Enter your name: Mariia
Hello, Mariia
Для продолжения нажмите любую клавишу . . .
```

4. Данное консольное приложение я рассмотрела для изучения функций побитового сдвига. Мы рассматриваем число 6. В двоичной СС оно записывается как 110. Побитовый сдвиг влево (shl) на 1 = умножение на 2^1 . Побитовый сдвиг влево на 3 = умножение на 2^3 . Приложение выводит нам результат. ($6 * 2^3 = 6 * 8 = 48$)

 flat assembler 1.73.25

```
File Edit Search Run Options Help
format PE console 4.0
entry Start
include 'win32a.inc'
section '.data' data readable writeable

    resStr db 'Result: %d', 0

section '.code' code readable executable


Start:
    mov ecx, 110b
    shl ecx, 3

    push ecx
    push resStr
    call [printf]

    call [getch]


section '.idata' import data readable
library kernel, 'kernel32.dll', \
    msvcrt, 'msvcrt.dll'

import kernel, \
    ExitProcess, 'ExitProcess'
import msvcrt, \
    printf, 'printf', \
    scanf, 'scanf', \
    getch, '_getch'
```

 C:\Users\astaskina\Documents\FASM\trial.EXE

Result: 48

Побитовый сдвиг вправо (shr) на 1 = разделить на 2^1 . Приложение выводит результат: $6 / 2^1 = 6 / 2 = 3$.

 flat assembler 1.73.25

File Edit Search Run Options Help

```
format PE console 4.0
entry Start
include 'win32a.inc'
section '.data' data readable writeable

    resStr db 'Result: %d', 0

section '.code' code readable executable

    Start:
        mov ecx, 110b
        shr ecx, 1_

        push ecx
        push resStr
        call [printf]

        call [getch]

section '.idata' import data readable
library kernel, 'kernel32.dll', \
    msvcrt, 'msvcrt.dll'

import kernel, \
    ExitProcess, 'ExitProcess'
import msvcrt, \
    printf, 'printf', \
    scanf, 'scanf', \
    getch, '_getch'
```

C:\Users\astaskina\Documents\FASM\trial.EXE

Result: 3

5. Данное консольное приложение я рассмотрела для изучения различных логических функций. Логическое умножение (and). $6 \text{ and } 5 = 110b \text{ and } 101b = 100b = 4$.

flat assembler 1.73.25

File Edit Search Run Options Help

```
format PE console 4.0
entry Start
include 'win32a.inc'
section '.data' data readable writeable

    resStr db 'Result: %d', 0

section '.code' code readable executable

    Start:
        mov eax, 110b
        and eax, 101b

        push eax
        push resStr
        call [printf]

        call [getch]

section '.idata' import data readable
library kernel, 'kernel32.dll', \
    msvcrt, 'msvcrt.dll'

import kernel, \
    ExitProcess, 'ExitProcess'
import msvcrt, \
    printf, 'printf', \
    scanf, 'scanf', \
    getch, '_getch'
```

C:\Users\astaskina\Documents\FASM\trial.EXE

Result: 4_

Логическое сложение (or). $8 \text{ or } 10 = 1000b \text{ or } 1010b = 1010b = 10$.

```
format PE console 4.0
entry Start
include 'win32a.inc'
section '.data' data readable writeable

    resStr db 'Result: %d', 0

section '.code' code readable executable


    Start:
        mov eax, 1000b
        or eax, 1010b

        push eax
        push resStr
        call [printf]

        call [getch]

section '.idata' import data readable
library kernel, 'kernel32.dll', \
    msvcrt, 'msvcrt.dll'

import kernel, \
    ExitProcess, 'ExitProcess'
import msvcrt, \
    printf, 'printf', \
    scanf, 'scanf', \
    getch, '_getch'
```

 C:\Users\astaskina\Documents\FASM\trial.EXE

Result: 10

Побитовое сложение (xor). $10 \text{ xor } 7 = 1010b \text{ xor } 111b = 1101b = 13$.


```
format PE console 4.0
entry Start
include 'win32a.inc'
section '.data' data readable writeable

    resStr db 'Result: %d', 0

section '.code' code readable executable


    Start:
        mov eax, 1010b
        xor eax, 111b

        push eax
        push resStr
        call [printf]

        call [getch]

section '.idata' import data readable
library kernel, 'kernel32.dll', \
    msvcrt, 'msvcrt.dll'

import kernel, \
    ExitProcess, 'ExitProcess'
import msvcrt, \
    printf, 'printf', \
    scanf, 'scanf', \
    getch, '_getch'
```

 C:\Users\astaskina\Documents\FASM\trial.EXE

Result: 13