

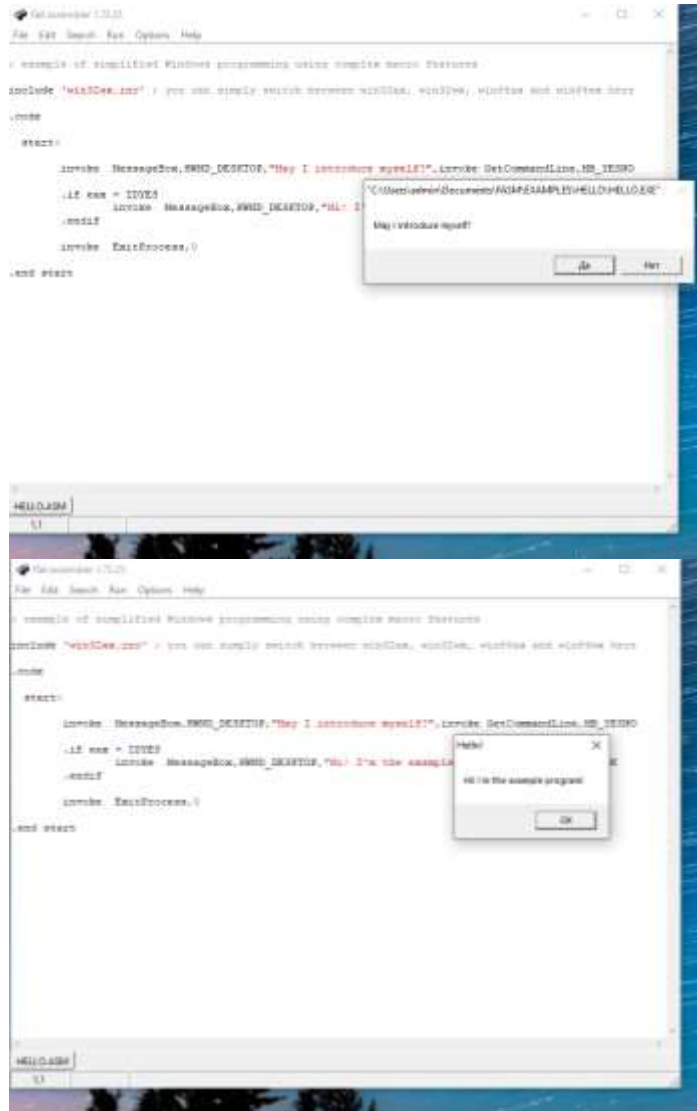
Семинар №1. Отчет о проделанной работе.

Филиппова Мария, БПИ-196

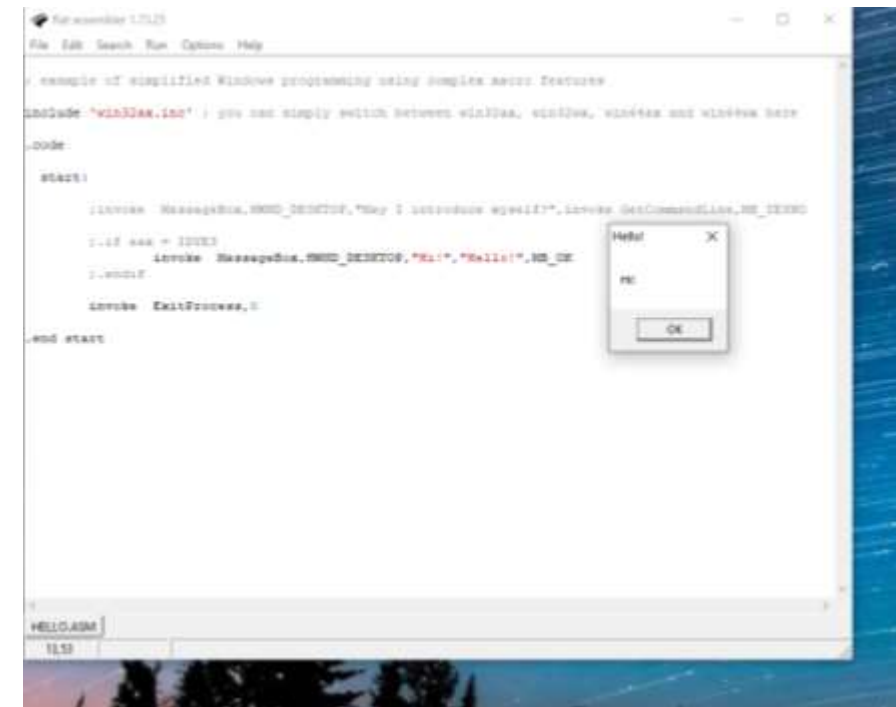
Задание

Найти в различных источниках информации не менее пяти примеров консольных программ разной сложности, написанных на изучаемом Ассемблере. Откомпилировать и запустить их с различными тестовыми наборами данных. Результаты работы программ зафиксировать в виде скриншотов.

1. Вывод на экран



Изменим программу: закоментируем конструкцию if



2. Калькулятор

$3 - 3 = 0$

```
format PE console

entry Start

.include 'win32a.inc'

section '.data' data readable writable
    strA db 'Enter A: ', 0
    strB db 'Enter B: ', 0
    strOp db 'Enter operation: ', 0

    resStr db 'Result: %d', 0
    resMod db '/%d', 0

    spaceStr db ' ', 0
    emptyStr db '%d', 0

    infinity db 'infinity', 0
    point db ',', 0

    A dd ?
    B dd ?
    C dd ?

    NULL = 0

section '.code' code readable executable

    Start:
        push strA
        call [printf]

        push A
        push spaceStr
```

$2 / 0 = \text{infinity}$

```
format PE console

entry Start

.include 'win32a.inc'

section '.data' data readable writable
    strA db 'Enter A: ', 0
    strB db 'Enter B: ', 0
    strOp db 'Enter operation: ', 0

    resStr db 'Result: %d', 0
    resMod db '/%d', 0

    spaceStr db ' ', 0
    emptyStr db '%d', 0

    infinity db 'infinity', 0
    point db ',', 0

    A dd ?
    B dd ?
    C dd ?

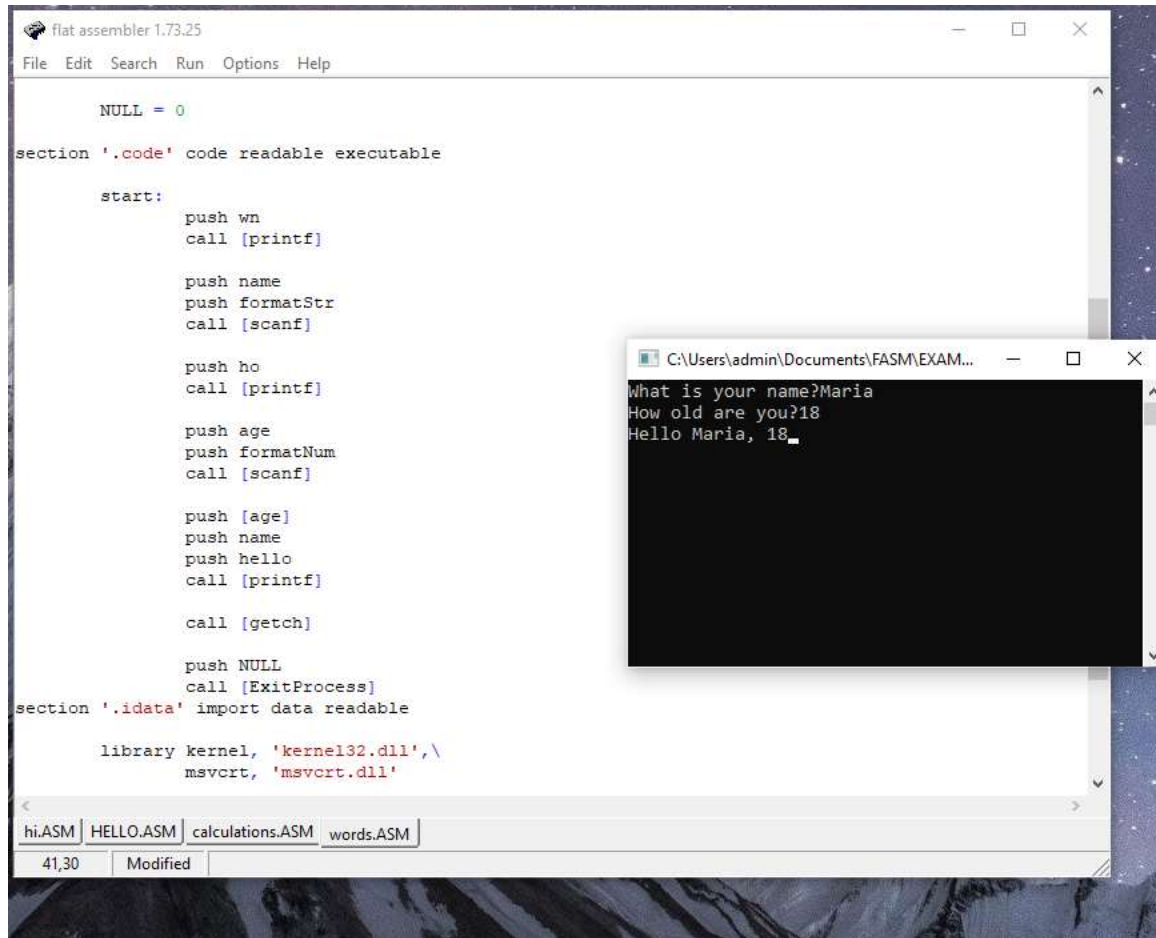
    NULL = 0

section '.code' code readable executable

    Start:
        push strA
        call [printf]

        push A
        push spaceStr
```

3. «Вежливое» приложение



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

NULL = 0

section '.code' code readable executable

start:
    push wn
    call [printf]

    push name
    push formatStr
    call [scanf]

    push ho
    call [printf]

    push age
    push formatNum
    call [scanf]

    push [age]
    push name
    push hello
    call [printf]

    call [getch]

    push NULL
    call [ExitProcess]

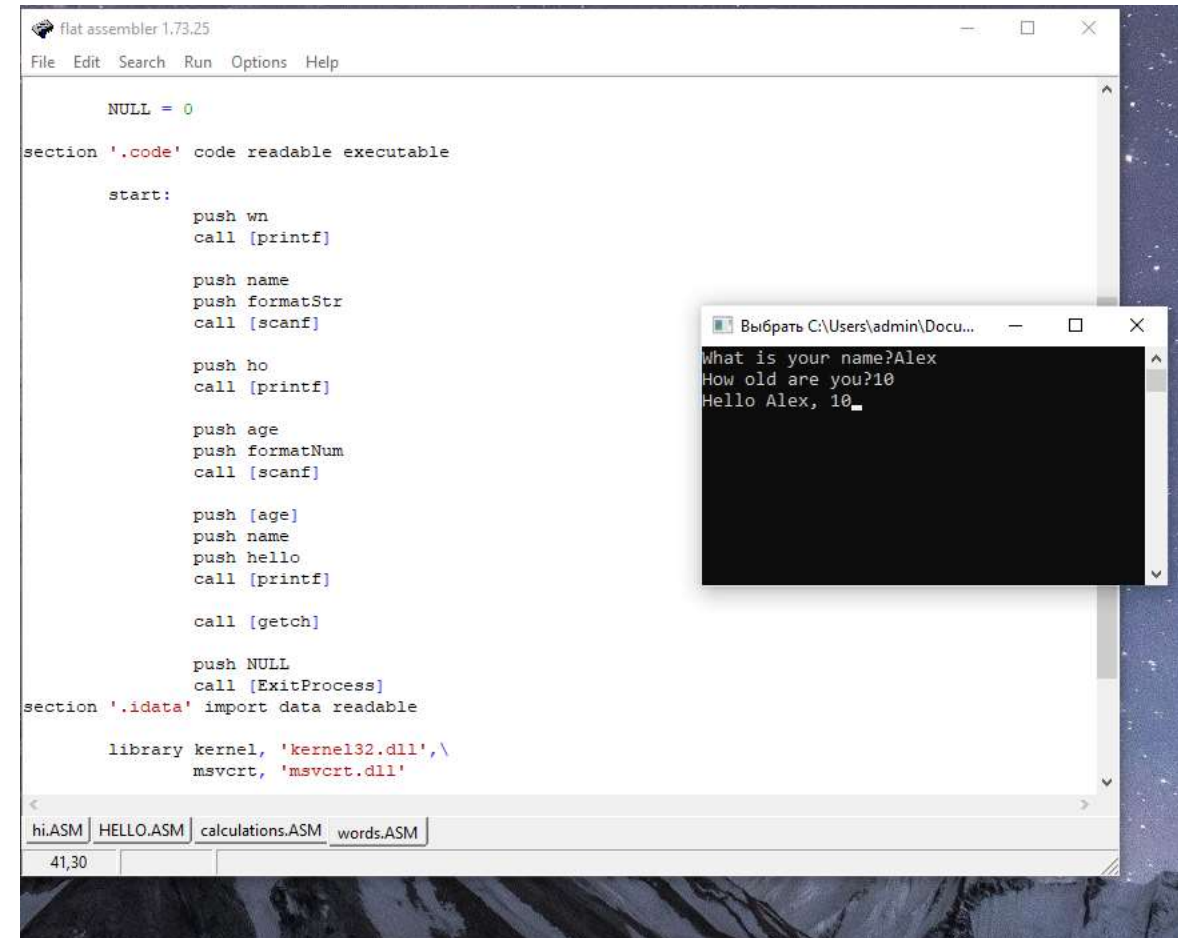
section '.idata' import data readable

library kernel, 'kernel32.dll', \
    msvcrt, 'msvcrt.dll'

hi.ASM HELLO.ASM calculations.ASM words.ASM
41,30 Modified
```

Console output:

```
C:\Users\admin\Documents\FASM\EXAM...
what is your name?Maria
How old are you?18
Hello Maria, 18_
```



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

NULL = 0

section '.code' code readable executable

start:
    push wn
    call [printf]

    push name
    push formatStr
    call [scanf]

    push ho
    call [printf]

    push age
    push formatNum
    call [scanf]

    push [age]
    push name
    push hello
    call [printf]

    call [getch]

    push NULL
    call [ExitProcess]

section '.idata' import data readable

library kernel, 'kernel32.dll', \
    msvcrt, 'msvcrt.dll'

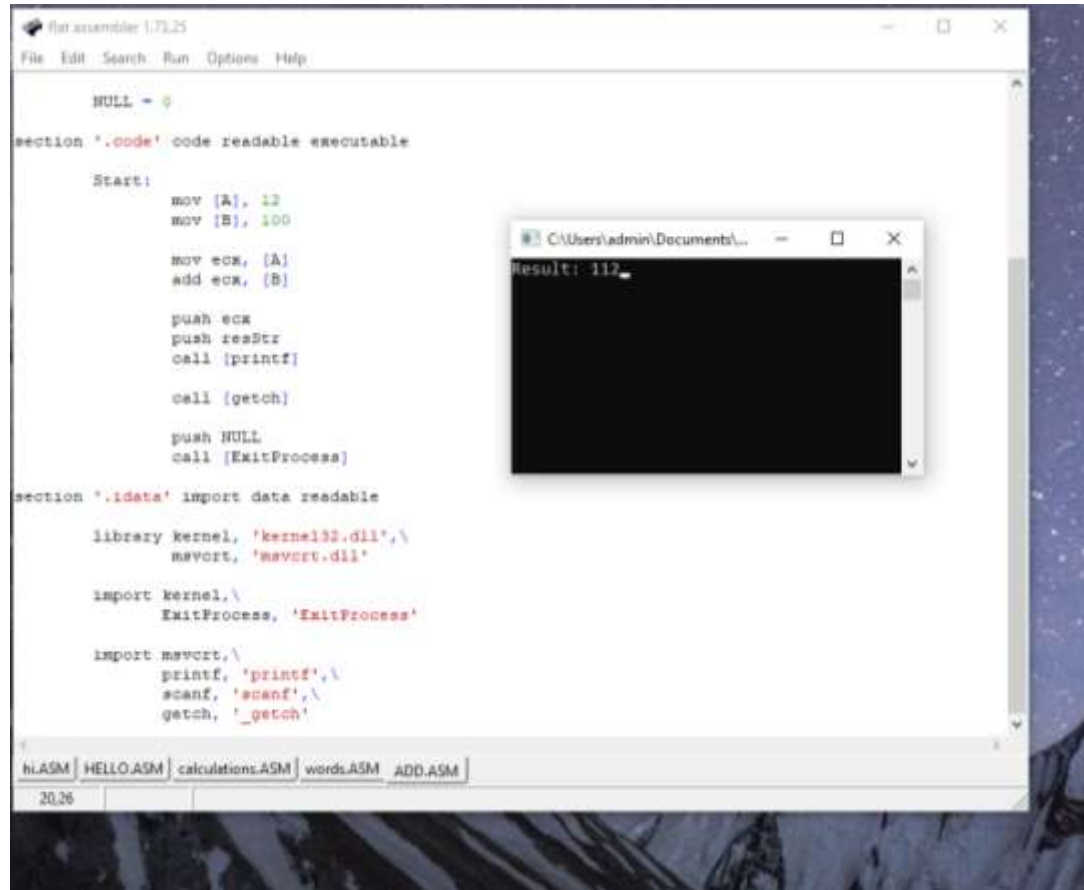
hi.ASM HELLO.ASM calculations.ASM words.ASM
41,30
```

Console output:

```
Выбрать C:\Users\admin\Docu...
what is your name?Alex
How old are you?10
Hello Alex, 10_
```

4. Сложение 2 чисел

$$12 + 100 = 112$$



```

NULL = 0

section '.code' code readable executable

Start:
    mov [A], 12
    mov [B], 100

    mov ecx, [A]
    add ecx, [B]

    push ecx
    push readStr
    call [printf]

    call [getch]

    push NULL
    call [ExitProcess]

section '.idata' import data readable

library kernel, 'kernel32.dll', \
    user32, 'user32.dll'

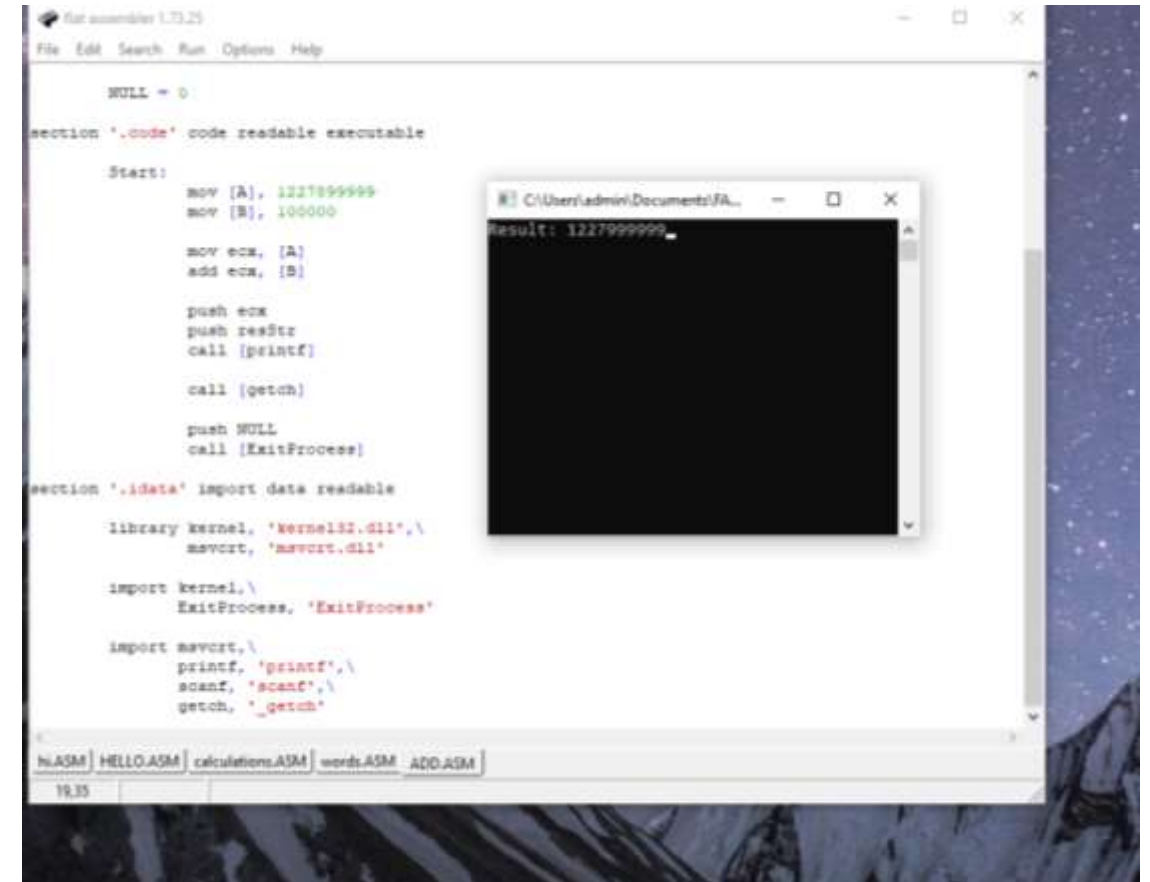
import kernel, \
    ExitProcess, 'ExitProcess'

import user32, \
    printf, 'printf', \
    scanf, 'scanf', \
    getch, '_getch'


```

Result: 112

$$1227899999 + 100000 = 1227999999$$



```

NULL = 0

section '.code' code readable executable

Start:
    mov [A], 1227899999
    mov [B], 100000

    mov ecx, [A]
    add ecx, [B]

    push ecx
    push readStr
    call [printf]

    call [getch]

    push NULL
    call [ExitProcess]

section '.idata' import data readable

library kernel, 'kernel32.dll', \
    user32, 'user32.dll'

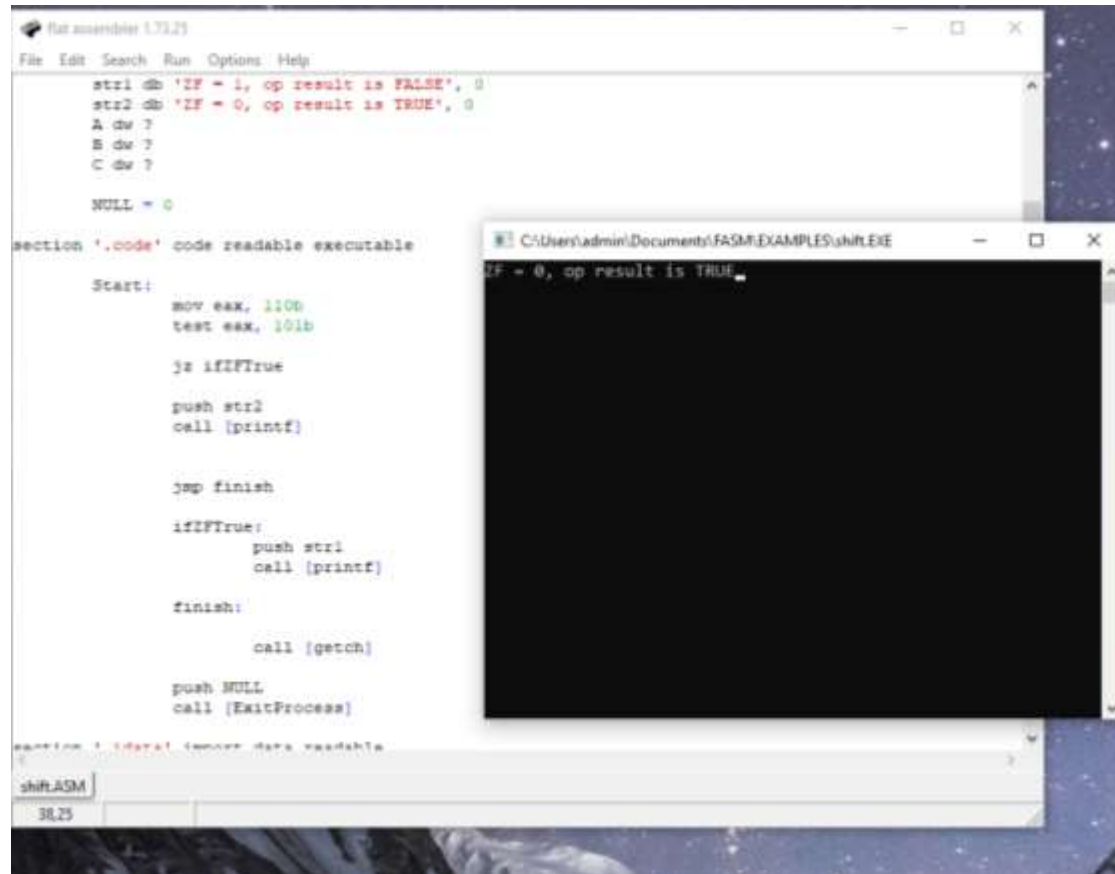
import kernel, \
    ExitProcess, 'ExitProcess'

import user32, \
    printf, 'printf', \
    scanf, 'scanf', \
    getch, '_getch'


```

Result: 1227999999

5. Логическая команда test



The screenshot shows the Flat assembler 1.73.25 interface. The assembly code defines two strings, `str1` and `str2`, and three variables `A`, `B`, and `C`. The `section '.code'` contains the main logic. It starts by moving `110b` into `eax` and testing it with `101b`. If the result is false (`js ifIfTrue`), it prints `str2` and jumps to `finish`. If true, it prints `str1`. The `finish` label calls `getch` and `ExitProcess`. A console window titled `C:\Users\admin\Documents\FASM\EXAMPLES\shift.EXE` shows the output: `IF = 0, op result is TRUE`.

```
str1 db 'IF = 1, op result is FALSE', 0
str2 db 'IF = 0, op result is TRUE', 0
A dw ?
B dw ?
C dw ?

NULL = 0

section '.code' code readable executable

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

    js ifIfTrue

    push str2
    call [printf]

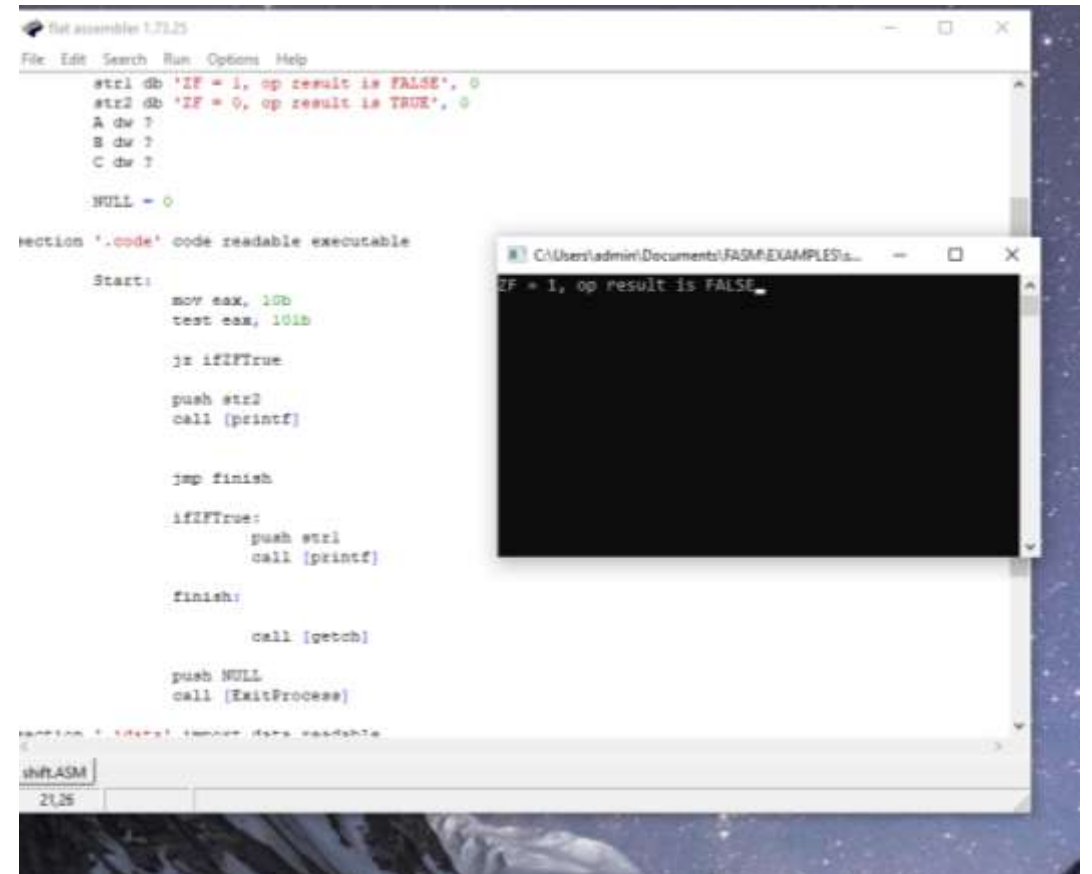
    jmp finish

ifIfTrue:
    push str1
    call [printf]

finish:
    call [getch]

    push NULL
    call [ExitProcess]
```

shift_ASM
38,25



The screenshot shows the Flat assembler 1.73.25 interface with the same assembly code as the previous image. However, the console window titled `C:\Users\admin\Documents\FASM\EXAMPLES\shift.EXE` shows the output: `IF = 1, op result is FALSE`. This indicates that the value in `eax` was not zero, leading to the execution of the `ifIfTrue` block.

```
str1 db 'IF = 1, op result is FALSE', 0
str2 db 'IF = 0, op result is TRUE', 0
A dw ?
B dw ?
C dw ?

NULL = 0

section '.code' code readable executable

Start:
    mov eax, 10b
    test eax, 101b

    js ifIfTrue

    push str2
    call [printf]

    jmp finish

ifIfTrue:
    push str1
    call [printf]

finish:
    call [getch]

    push NULL
    call [ExitProcess]
```

shift_ASM
21,25