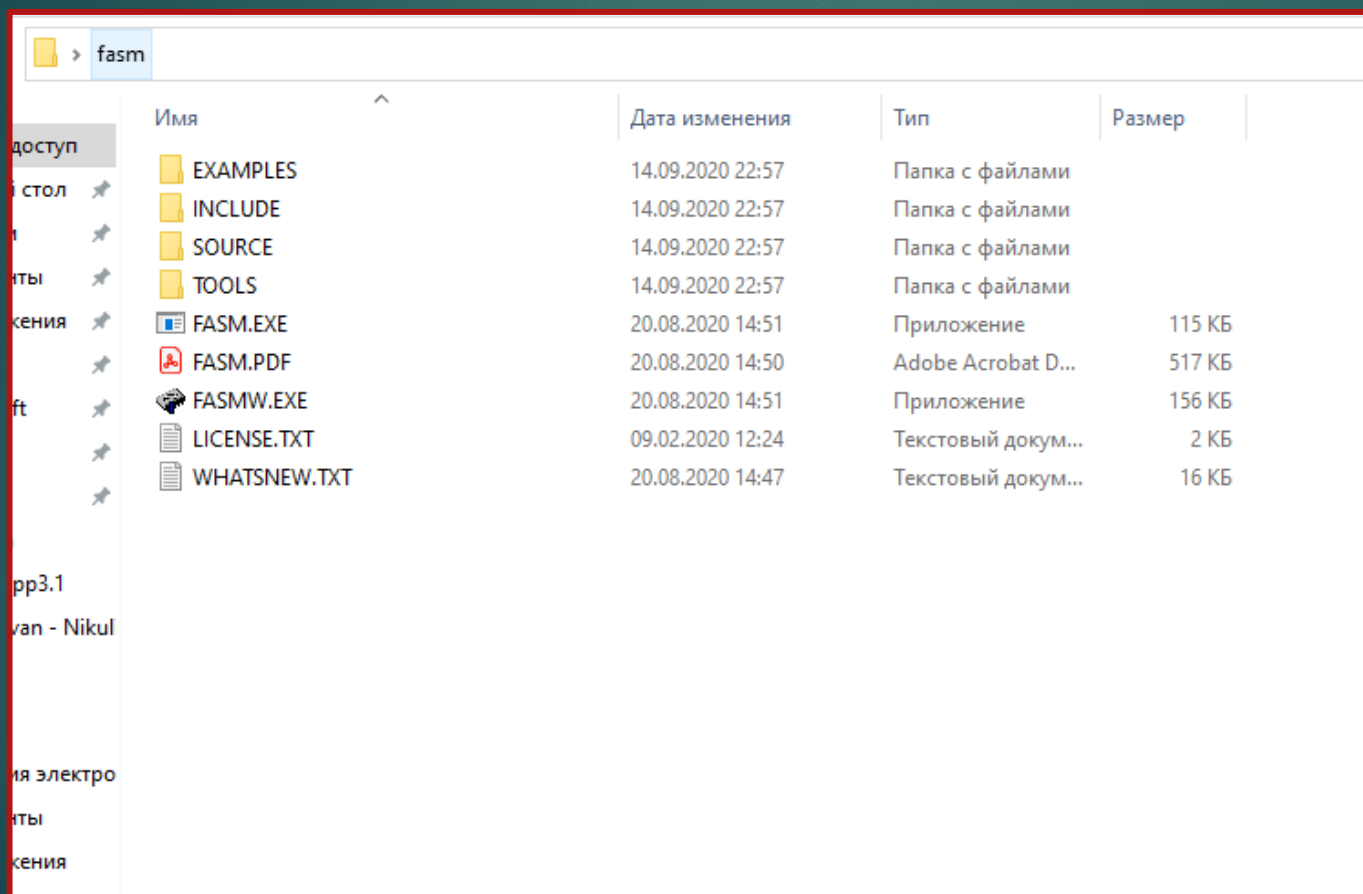


АВС ДЗ1: Освоение среды FASM

МАХНАЧ ФЁДОР, БПИ196

УСТАНОВКА

- ▶ Загрузил архив с <https://flatassembler.net/download.php>, распаковал в папку на рабочем столе.



Первый пример

- ▶ Запустил FASMW.exe.
В первую очередь решил выбрать для рассмотрения один из приложенных к FASM-у примеров: PEDEMO.ASM.
- ▶ Видим, что это программа с GUI (format PE GUI), притом небольшая. Строка Hello World наталкивает на мысль, что это будет окно с соответствующей надписью.

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

; Example of making 32-bit PE program as raw code and data

format PE GUI
entry start

section '.text' code readable executable

    start:

        push    0
        push    _caption
        push    _message
        push    0
        call    [MessageBoxA]

        push    0
        call    [ExitProcess]

section '.data' data readable writeable

    _caption db 'Win32 assembly program',0
    _message db 'Hello World!',0

section '.idata' import data readable writeable

dd 0,0,0,RVA kernel_name,RVA kernel_table
dd 0,0,0,RVA user_name,RVA user_table
dd 0,0,0,0,0

kernel_table:
    ExitProcess dd RVA _ExitProcess
    dd 0
user_table:
    MessageBoxA dd RVA _MessageBoxA
    dd 0

kernel_name db 'KERNEL32.DLL',0
user_name db 'USER32.DLL',0

_ExitProcess dw 0
db 'ExitProcess',0
_MessageBoxA dw 0
db 'MessageBoxA',0

section '.reloc' fixups data readable discardable    ; needed for Win32s
```

Первый пример

- ▶ Запуск программы (Run -> Run) подтверждает предположение.
- ▶ То, что строки в секции .data относятся к надписям на всплывающем окне не вызывает сомнений. Интересен вызов функции – сначала «толкаются» значения, потом вызывается функция. Видимо, если поменять местами 2 команды push, надпись и заголовок всплывающего окна поменяются местами.

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

; Example of making 32-bit PE program as raw code and data

format PE GUI
entry start

section '.text' code readable executable

start:

    push    0
    push    _caption
    push    _message
    push    0
    call    [MessageBoxA]

    push    0
    call    [ExitProcess]

section '.data' data readable writeable

_caption db 'Win32 assembly program',0
_message db 'Hello World!',0

section '.idata' import data readable writeable

dd 0,0,0,RVA kernel_table
dd 0,0,0,RVA user_table
dd 0,0,0,0,0

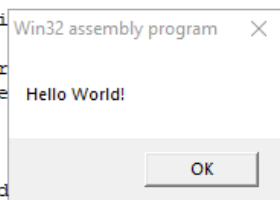
kernel_table:
    ExitProcess dd 0
    dd 0

user_table:
    MessageBoxA dd RVA _MessageBoxA
    dd 0

kernel_name db 'KERNEL32.DLL',0
user_name db 'USER32.DLL',0

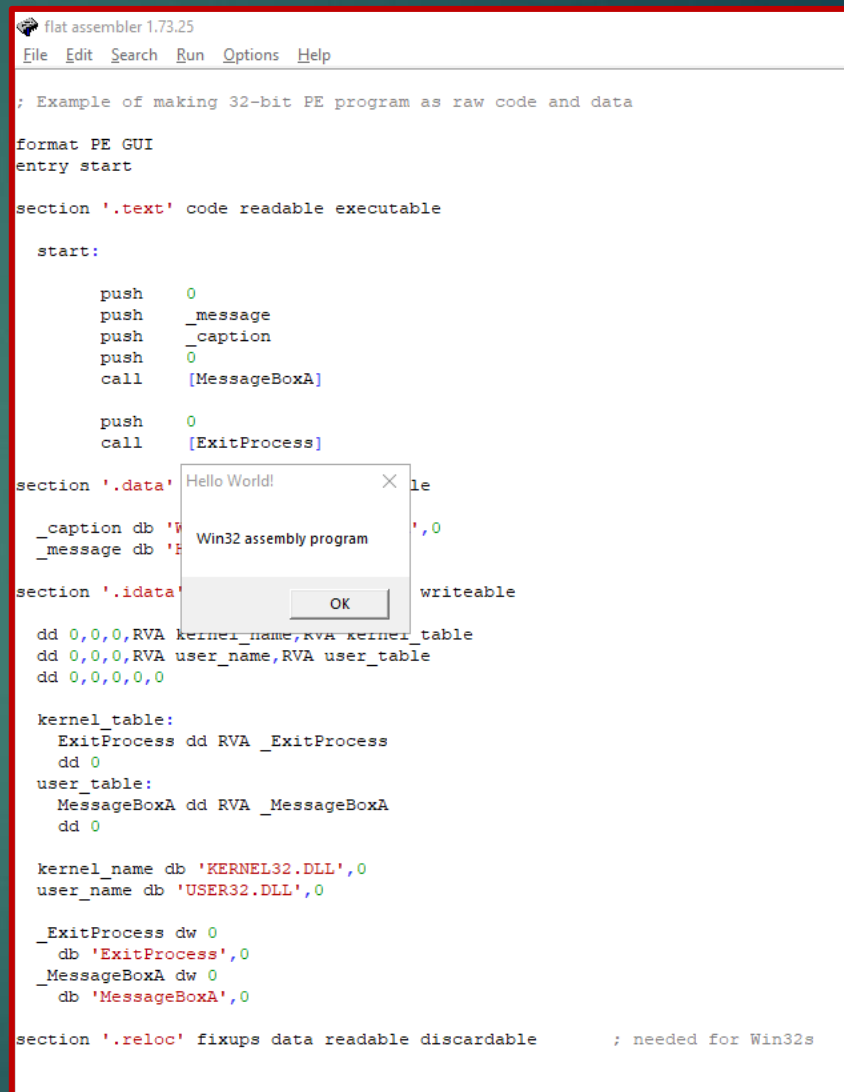
_ExitProcess dw 0
    db 'ExitProcess',0
_MessageBoxA dw 0
    db 'MessageBoxA',0

section '.reloc' fixups data readable discardable ; needed for Win32s
```



Первый пример

- ▶ Компилирую и запускаю код, предположение подтвердилось.
- ▶ Возник вопрос, зачем толкаются 2 нуля для MessageBoxA.
- ▶ С первым нулём что-то вышло – при изменении изменялся тип окошка (см. след. слайд). Изменение второго приводило к неработоспособности программы.
- ▶ Погуглил функцию, оказалось, это ссылка на owner window.
- ▶ Стоит отметить, что параметры push-атся в обратном порядке (логично, чтобы потом считать в прямом со стека).



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

; Example of making 32-bit PE program as raw code and data

format PE GUI
entry start

section '.text' code readable executable

start:

    push    0
    push    _message
    push    _caption
    push    0
    call    [MessageBoxA]

    push    0
    call    [ExitProcess]

section '.data' data readable writeable
    _caption db 'Win32 assembly program',0
    _message db 'Hello World!',0

section '.idata' import data readable writeable
dd 0,0,0,RVA kernel_name,RVA kernel_table
dd 0,0,0,RVA user_name,RVA user_table
dd 0,0,0,0,0

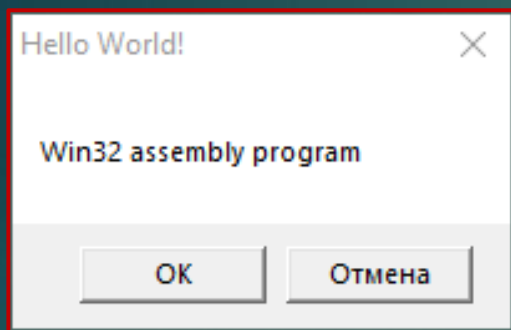
kernel_table:
    ExitProcess dd RVA _ExitProcess
    dd 0
user_table:
    MessageBoxA dd RVA _MessageBoxA
    dd 0

kernel_name db 'KERNEL32.DLL',0
user_name db 'USER32.DLL',0

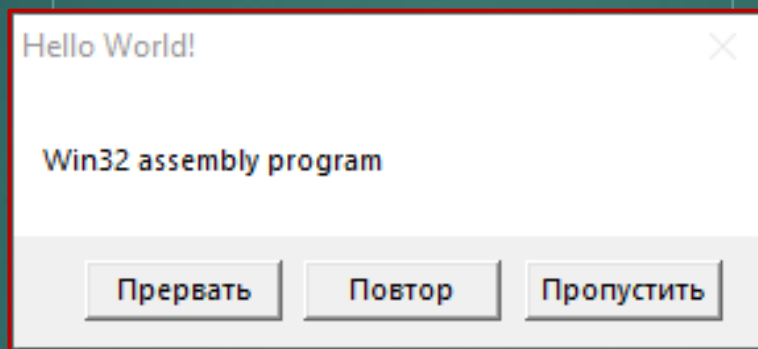
_ExitProcess dw 0
    db 'ExitProcess',0
_MessageBoxA dw 0
    db 'MessageBoxA',0

section '.reloc' fixups data readable discardable ; needed for Win32s
```

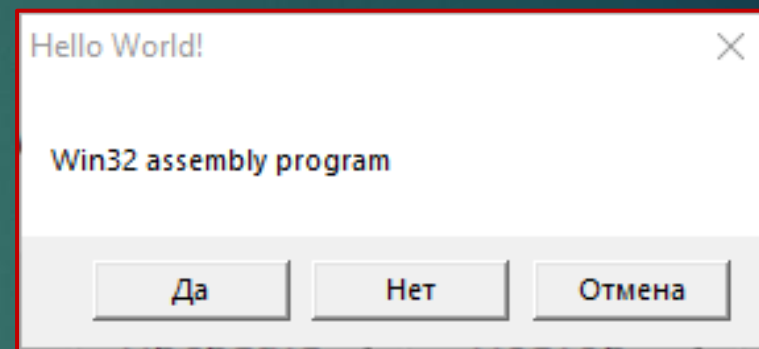
Первый пример



push 1



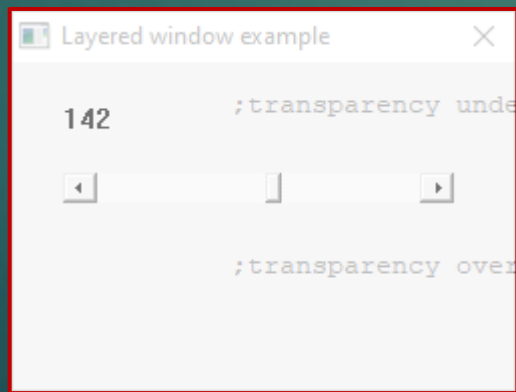
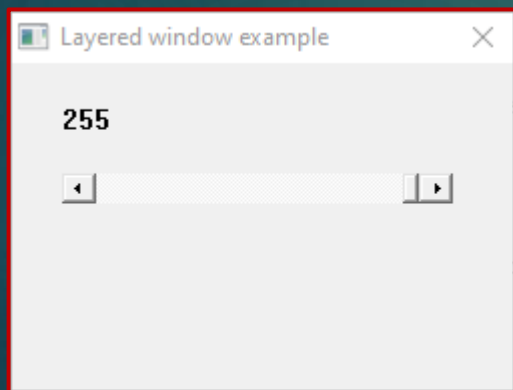
push 2



push 3

Второй пример

- ▶ Другой пример программы нашёл на <https://flatassembler.net/examples.php>. Программа выводит окно, прозрачность которого можно изменять (layeredwin.asm).



```
flat assembler 1.73.25
File Edit Search Run Options Help
;setting window transparency example by carlos hernandez/coconut

format PE GUI 4.0
entry start

ID_SCROLLBAR = 1000
ID_STATIC = 1001

include 'win32w.inc'

section '.data' data readable writeable

    _class TCHAR 'FASMIN32',0
    _title TCHAR 'Layered window example',0
    _error TCHAR 'Startup failed.',0
    _scrollbar TCHAR 'scrollbar',0
    _static TCHAR 'static',0
    _conv TCHAR '%i',0

    hwndstatic dd ?
    hwndscroll dd ?

    buffer rb 32

    wc WNDCLASS 0,WindowProc,0,0,NULL,NULL,NULL,COLOR_BTNFACE+1,NULL,_class

    lpsi SCROLLINFO sizeof.SCROLLINFO,SIF_ALL,5,255,1,255,0

    msg MSG

section '.code' code readable executable

start:
    invoke GetModuleHandle,0
    mov [wc.hInstance],eax
    invoke LoadIcon,0,IDI_APPLICATION
    mov [wc.hIcon],eax
    invoke LoadCursor,0,IDC_ARROW
    mov [wc.hCursor],eax
    invoke RegisterClass,wc
    or eax,eax
    jz error
    invoke CreateWindowEx,WS_EX_LAYERED,_class,_title,WS_VISIBLE+WS_SYSMENU,128,128,256,192,NULL,NULL,[wc.hInstance],NULL
    test eax,eax
    jz error

msg_loop:
    invoke GetMessage,msg,NULL,0,0
    or eax,eax
    jz end_loop
    invoke TranslateMessage,msg
    invoke DispatchMessage,msg
    jmp msg_loop

error:
    invoke MessageBox,NULL,_error,NULL,MB_ICONERROR+MB_OK
end_loop:
    invoke ExitProcess,[msg.wParam]
```


Третий пример

- ▶ hello.asm
- ▶ Источник: github.com/macton
- ▶ Очень много кода, но выводит всего лишь Hello world в диалоговом окне.
- ▶ Предположительно, преимущество заключается в размере исполняемого файла (в 2 раза меньше, чем PEDEMO.exe).

C:\Users\fedya\OneDrive\Рабочий стол\fasn\fasn_projs\task1\hell... X

Hello, World!

OK

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

IMAGE_DIRECTORY_ENTRY_GLOBALPTR:

    dd 0 ; u32 VirtualAddress
    dd 0 ; u32 Size

IMAGE_DIRECTORY_ENTRY_TLS:

    dd 0 ; u32 VirtualAddress
    dd 0 ; u32 Size

IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG:

    dd 0 ; u32 VirtualAddress
    dd 0 ; u32 Size

IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT:

    dd 0 ; u32 VirtualAddress
    dd 0 ; u32 Size

IMAGE_DIRECTORY_ENTRY_IAT:

    dd 0 ; u32 VirtualAddress
    dd 0 ; u32 Size

IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT:

    dd 0 ; u32 VirtualAddress
    dd 0 ; u32 Size

IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR:

    dd 0 ; u32 VirtualAddress
    dd 0 ; u32 Size

IMAGE_DIRECTORY_ENTRY_RESERVED:

    dd 0 ; u32 VirtualAddress
    dd 0 ; u32 Size

IMAGE_DATA_DIRECTORIES_END:
IMAGE_DATA_DIRECTORIES_COUNT = (IMAGE_DATA_DIRECTORIES_END-IMAGE_DATA_DIRECTORIES)/8

IMAGE_OPTIONAL_HEADER_END:
IMAGE_OPTIONAL_HEADER_SIZE = IMAGE_OPTIONAL_HEADER_END-IMAGE_OPTIONAL_HEADER
assert (IMAGE_OPTIONAL_HEADER_SIZE = 0xF0)

SECTION TABLE:
; Characteristics
.IMAGE_SCN_TYPE_REG                = 0x00000000
.IMAGE_SCN_TYPE_DSECT              = 0x00000001
.IMAGE_SCN_TYPE_NOLOAD             = 0x00000002
.IMAGE_SCN_TYPE_GROUP              = 0x00000004
.IMAGE_SCN_TYPE_NO_PAD             = 0x00000008
.IMAGE_SCN_TYPE_COPY               = 0x00000010
.IMAGE_SCN_CNT_CODE                = 0x00000020
.IMAGE_SCN_CNT_INITIALIZED_DATA    = 0x00000040
.IMAGE_SCN_CNT_UNINITIALIZED_DATA  = 0x00000080

<
hello.ASM
1,1
```


Четвёртый пример

```
flat assembler 1.73.25
File Edit Search Run Options Help
; 01_pe_printf_01.asm
; - Named flags for section table characteristics
; - Add BSS section (note exe file size does not change)
; - Write data into bss (my_value)
; - Verify by reading data from bss and printing values

format binary as "exe"
org 0
use64

SECTION_ALIGNMENT = 0x00001000
FILE_ALIGNMENT    = 0x00000200
IMAGE_BASE        = 0x0000000000040000
STACK_RESERVE_SIZE = 0x0000000000000100
STACK_COMMIT_SIZE  = 0x0000000000000100
HEAP_RESERVE_SIZE  = 0x000000000000010000
HEAP_COMMIT_SIZE   = 0x000000000000000000
CODE_BASE          = CODE.RVA

macro print_value_x32 description, value
{
    bits = 32
    display description
    display '0x'
    repeat bits/4
        d = '0' + value shr (bits-4) and 0Fh
        if d > '9'
            d = d + 'A'-'9'-1
        end if
        display d
    end repeat
    display $a
}

macro align_section
{
    db (((($+(FILE_ALIGNMENT-1))/FILE_ALIGNMENT)*FILE_ALIGNMENT)-$) dup (0)
}

IMAGE_DOS_HEADER:

db 0x4d, 0x5a          ; ul6_e_magic
db 0x80, 0x00          ; ul6_e_cblp
db 0x01, 0x00          ; ul6_e_cp
db 0x00, 0x00          ; ul6_e_crlc
db 0x04, 0x00          ; ul6_e_cparhdr
db 0x10, 0x00          ; ul6_e_ss
db 0xff, 0xff          ; ul6_e_maxalloc
db 0x00, 0x00          ; ul6_e_minalloc
db 0x40, 0x01          ; ul6_e_sp
db 0x00, 0x00          ; ul6_e_csum
db 0x00, 0x00          ; ul6_e_ip
db 0x00, 0x00          ; ul6_e_cs
db 0x40, 0x00          ; ul6_e_lfarlc
db 0x00, 0x00          ; ul6_e_ovno
db 0x00, 0x00          ; ul6_e_res[0]
db 0x00, 0x00          ; ul6_e_res[1]
db 0x00, 0x00          ; ul6_e_res[2]
db 0x00, 0x00          ; ul6_e_res[3]
```

- ▶ 01_pe_printf_01.asm
- ▶ Источник: github.com/macton
- ▶ Записывает числа в bss секцию и выводит в консоль.

Четвёртый пример

```
C:\Users\fedya\OneDrive\Рабочий стол\fasn\fasn_projs\task1\test.exe
00000000004048F0: my_value[798] = 1596
00000000004048F8: my_value[799] = 1598
0000000000404900: my_value[800] = 1600
0000000000404908: my_value[801] = 1602
0000000000404910: my_value[802] = 1604
0000000000404918: my_value[803] = 1606
0000000000404920: my_value[804] = 1608
0000000000404928: my_value[805] = 1610
0000000000404930: my_value[806] = 1612
0000000000404938: my_value[807] = 1614
0000000000404940: my_value[808] = 1616
0000000000404948: my_value[809] = 1618
0000000000404950: my_value[810] = 1620
0000000000404958: my_value[811] = 1622
0000000000404960: my_value[812] = 1624
0000000000404968: my_value[813] = 1626
0000000000404970: my_value[814] = 1628
0000000000404978: my_value[815] = 1630
0000000000404980: my_value[816] = 1632
0000000000404988: my_value[817] = 1634
0000000000404990: my_value[818] = 1636
0000000000404998: my_value[819] = 1638
00000000004049A0: my_value[820] = 1640
00000000004049A8: my_value[821] = 1642
00000000004049B0: my_value[822] = 1644
00000000004049B8: my_value[823] = 1646
00000000004049C0: my_value[824] = 1648
00000000004049C8: my_value[825] = 1650
00000000004049D0: my_value[826] = 1652
```

Пятый пример

- ▶ Последний пример взял из [видеоурока](#) с канала Byte++.
- ▶ Программа (console.asm) запрашивает имя, возраст пользователя и выводит приветствие.
- ▶ Также в целях демонстрации работы команды lea программа выводит адрес одной из строк.

```
format PE console

entry start

include 'win32a.inc'

section '.data' data readable writable

    formatStr db '%s', 0
    formatNum db '%d', 0

    name rd 2
    age rd 1

    wn db 'What is your name? ', 0
    ho db 'How old are you? ', 0
    hello db 'Hello %s, %d', 0

    address db 10, 13, 'hello var address is %d', 0

    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]

    lea eax, [hello]

    push eax
    push address
    call [printf]

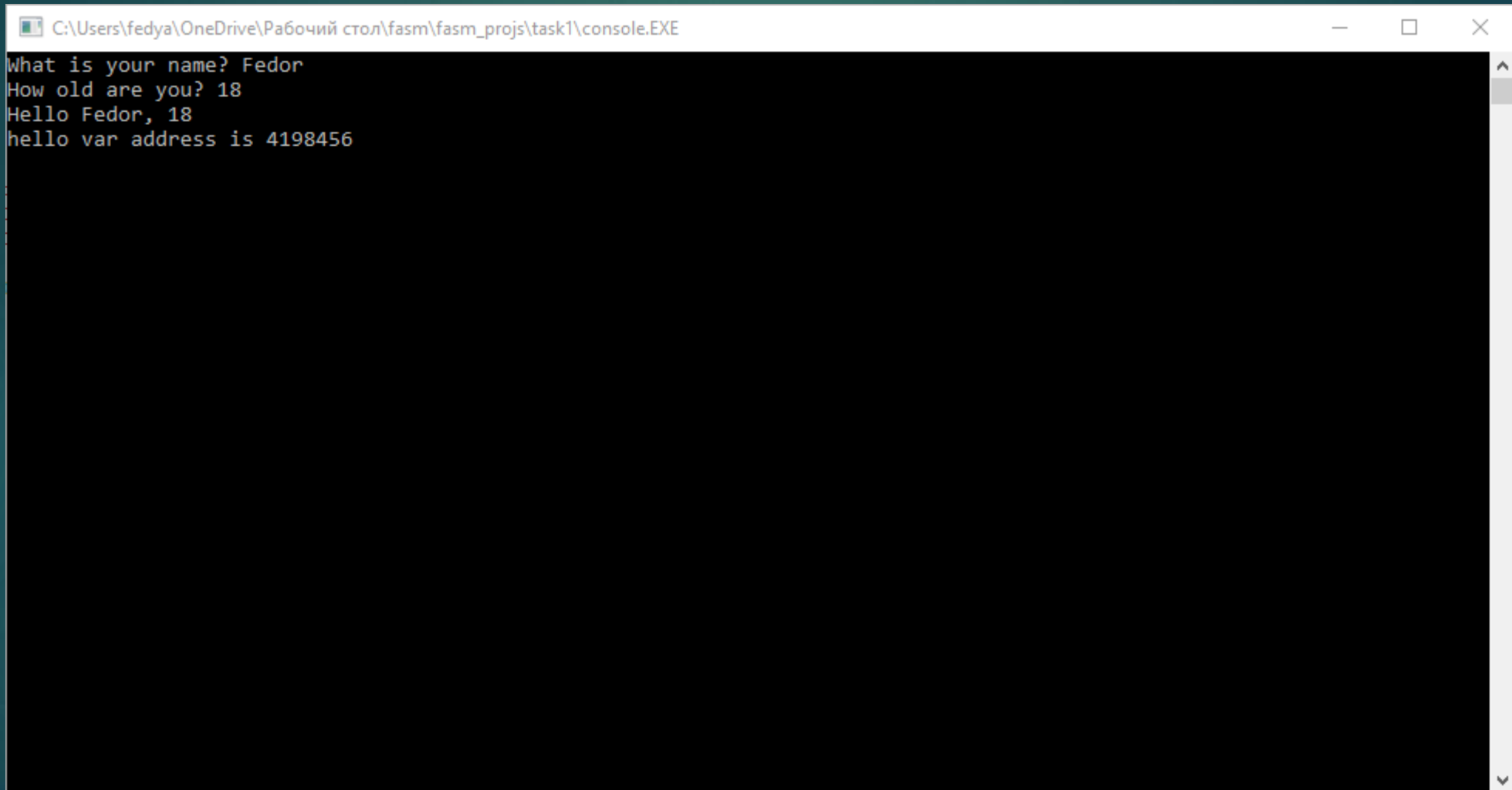
    call [getch]

    push NULL
    call [ExitProcess]

section '.idata' import data readable

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

Пятый пример



A screenshot of a Windows console window. The title bar shows the file path: C:\Users\fedya\OneDrive\Рабочий стол\iasm\iasm_projs\task1\console.EXE. The console output is as follows:

```
What is your name? Fedor
How old are you? 18
Hello Fedor, 18
hello var address is 4198456
```