**Федеральное государственное образовательное бюджетное учреждение  
высшего образования**

**«Финансовый университет при Правительстве Российской Федерации»**

**КОЛЛЕДЖ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ**

|  |  |
| --- | --- |
|  | |
|  |  |

**ОТЧЕТ**

**по практическому занятию**

**Машинно-ориентированное программирование для решения задач защиты информации**

**Студент:** Матвейчук Анастасия Сергеевна

**Специальность: 10.02.05 Обеспечение информационной безопасности автоматизированных систем**

**Группа: *3ОИБАС-718***

Оценка:\_\_\_\_\_\_\_\_\_\_\_\_

**Москва**

**2021 г.**

**Содержание**

[Введение 3](#_Toc62039254)

[Практическая часть 3](#_Toc62039255)

# Введение

**Цель работы:** изучение логических команд и командных сдвигов.

# Практическая часть

Настраиваем запуск отладчика.

Создаем файл Matveychuk.ASM и Matveychuk.Prog, затем открываем их через NotePad++.

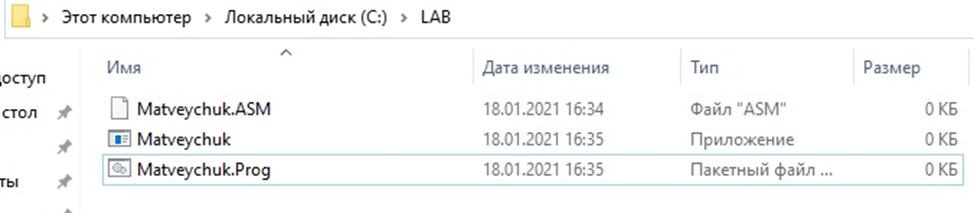


Рис. 1

В файле Matveychuk.ASM в первой строчке указываем путь к Fasm.exe и путь к нашему файлу Matveychuk.ASM.

Во второй строчке указываем путь, где необходимо создать EXE файл для запуска программы.

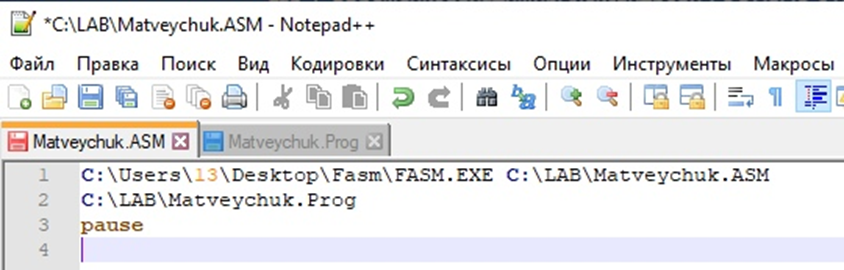


Рис. 2

Далее открываем файл Matveychuk.ASM.

В папку с программой закидываем скачанный файл \_\_Console\_V0.inc и прописываем пути.

В первой строчке пишем путь к WIN64A.INC. Во второй строчке пишем путь до \_\_Console\_V0.inc.



Рис. 3



Рис. 4

Переходим к написанию кода, вычисляющего сдвиги 32-х битных переменных, используя 16-ти битные операнды.

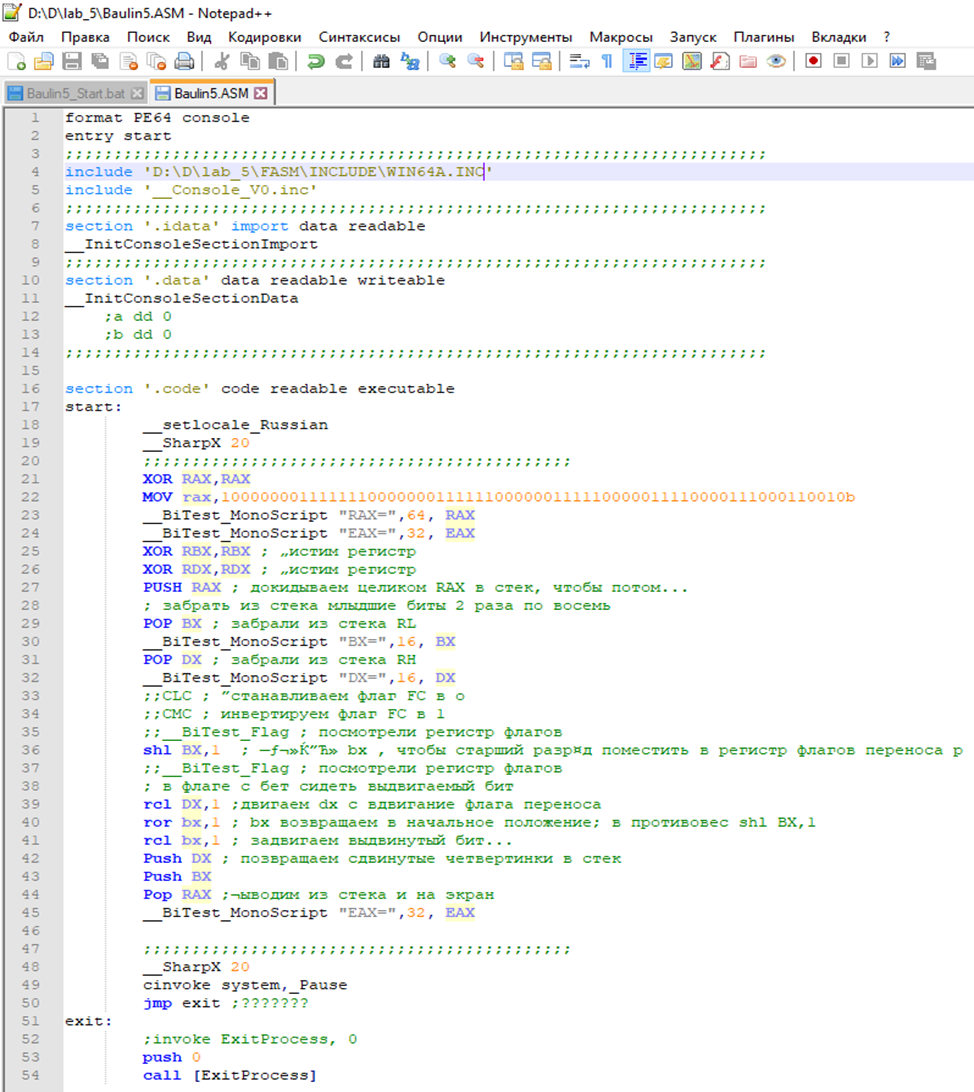


Рис. 5

Завершив, видим следующий результат.

Изображение выглядит как текст, снимок экрана, черный, экран

Автоматически созданное описание

Рис. 6

КОД ПРОГРАММЫ:

format PE64 console

entry start

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

include 'C:\C\LAB\FASM\INCLUDE\WIN64A.INC'

include '\_\_Console\_V0.inc'

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

section '.idata' import data readable

\_\_InitConsoleSectionImport

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

section '.data' data readable writeable

\_\_InitConsoleSectionData

;a dd 0

;b dd 0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

section '.code' code readable executable

start:

\_\_setlocale\_Russian

\_\_SharpX 20

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

XOR RAX,RAX

MOV rax,1000000011111110000000111111000000111110000011110000111000110010b

\_\_BiTest\_MonoScript "RAX=",64, RAX

\_\_BiTest\_MonoScript "EAX=",32, EAX

XOR RBX,RBX ; „истим регистр

XOR RDX,RDX ; „истим регистр

PUSH RAX ; докидываем целиком RAX в стек, чтобы потом...

; забрать из стека млыдшие биты 2 раза по восемь

POP BX ; забрали из стека RL

\_\_BiTest\_MonoScript "BX=",16, BX

POP DX ; забрали из стека RH

\_\_BiTest\_MonoScript "DX=",16, DX

;;CLC ; ”станавливаем флаг FC в о

;;CMC ; инвертируем флаг FC в 1

;;\_\_BiTest\_Flag ; посмотрели регистр флагов

shl BX,1 ; —ƒ¬»Ќ”Ћ» bx , чтобы старший разр¤д поместить в регистр флагов переноса р

;;\_\_BiTest\_Flag ; посмотрели регистр флагов

; в флаге с бет сидеть выдвигаемый бит

rcl DX,1 ;двигаем dx с вдвигание флага переноса

ror bx,1 ; bx возвращаем в начальное положение; в противовес shl BX,1

rcl bx,1 ; задвигаем выдвинутый бит...

Push DX ; позвращаем сдвинутые четвертинки в стек

Push BX

Pop RAX ;¬ыводим из стека и на экран

\_\_BiTest\_MonoScript "EAX=",32, EAX

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

\_\_SharpX 20

cinvoke system,\_Pause

jmp exit ;???????

exit:

;invoke ExitProcess, 0

push 0

call [ExitProcess]