МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

Лабораторная работа №1

по дисциплине: Архитектура вычислительных систем
тема: «Разработка программ на ассемблере.
Работа с отладчиком x32dbg, пакетом masm32»

Выполнил: ст. группы ПВ-233
Мовчан Антон Юрьевич

Проверили:
ст. пр. Осипов Олег Васильевич

Белгород 2025 г.

Лабораторная работа №1

Цель работы: получить навыки создания простейших ассемблерных программ с использованием пакета masm32 и научиться пользоваться отладчиком x32dbg.

Вариант 8

1. Ознакомиться со средой x32dbg и компилятором masm32.
2. Создать и скомпилировать программу в соответствии с вариантом задания. В программу включить комментарии с описанием, что делает каждая инструкция. Подробное описание каждой команды можно найти в приложении учебника В.И. Юрова «Assembler», начиная со стр. 511. Комментарии следует выровнять по левому краю (как в примере).
3. С помощью отладчика определить местонахождение переменных, строк и массивов в сегменте данных, а также их размер. Составить таблицу и подробное описание ячеек сегмента данных (как в примере).
4. Выполнить пошаговую трассировку программы. Определить какие регистры, флаги и ячейки памяти изменяют свои значения в процессе выполнения команд. Обеспечить корректное завершение программы вызовом системной функции ExitProcess с кодом завершения 0. Если в сегменте данных есть строки, то вывести её в консоль. Трассировку требуется выполнить до команды «call ExitProcess» включительно. Составить для каждой инструкции таблицу трассировки (как в примере).
5. Сделать выводы о проделанной работе.

Сегменты данных и кода имеют следующее содержание:

```
.DATA
          a DD 30201, 30201h
          b DB 43h, 0F3h, 0F3h, 0E5h
          DF 1500
          DD 1.5, 1.6, 1.9, -1.9
          t DQ 0E7D32A1h
          stra DB 16 dup(1)
     .CODE
     START:
          MOV ESI, 65737341h
          AND ESI, dword ptr b
          MOV dword ptr stra, ESI
          MOV ECX, dword ptr t
          IMUL ECX, 7
          ADD ECX, 6
          MOV dword ptr stra[4], ECX
          ADD stra[8], 'q'
          DEC stra[9]
     END START
```

Выполнение работы

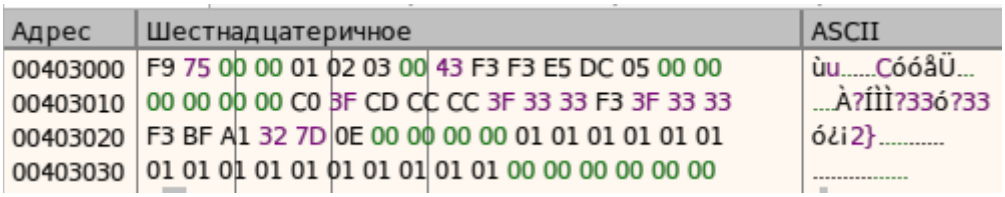
1. Создать файл lab1.asm со следующим содержанием:

```
.686
 .model flat, stdcall
 option casemap: none

 include windows.inc
 include kernel32.inc
 include msvcrt.inc
 includelib     kernel32.lib
 includelib     msvcrt.lib

.DATA
        a DD 30201, 30201h
        b DB 43h, 0F3h, 0F3h, 0E5h
        DF 1500
        DD 1.5, 1.6, 1.9, -1.9
        t DQ 0E7D32A1h
        stra DB 16 dup(1)
.CODE
START:
        MOV ESI, 65737341h              ; ESI = 65737341h
        AND ESI, dword ptr b            ; ESI = ESI & E5F3F343(т.к. DB просто определяет байты, а мы обращаемся к DD поэтому нужно развернуть)
        MOV dword ptr stra, ESI         ; stra[0:3] = ESI (41737365h)
        MOV ECX, dword ptr t            ; ECX = t[0:3] (0E7D32A1h)
        IMUL ECX, 7                     ; ECX = ECX * 7 (656C6267h)
        ADD ECX, 6                      ; ECX = ECX + 6 (656C626D)
        MOV dword ptr stra[4], ECX      ; stra[4:7] = ECX (6D626C65h)
        ADD stra[8], 'q'                ; stra[8] = 71h
        DEC stra[9]                     ; stra[9] -= 1 (0h)

        push offset stra
        call crt_puts                   ; puts(str1)
        ADD ESP, 4                      ; Очистка стека от аргумента

        call crt__getch                 ; Задержка ввода, getch()
        ; Вызов функции ExitProcess(0)
        push 0                          ; Поместить аргумент функции в стек
        call ExitProcess                ; Выход из программы
END START
```

2. Скомпилировать программу и получить исполняемый файл lab1.exe.

3. Открыть файл lab1.exe в отладчике.

4. Сегмент данных содержит три массива a, b, stra и переменную t



Адрес	Шестнадцатеричное	ASCII
00403000	F9 75 00 00 01 02 03 00 43 F3 F3 E5 DC 05 00 00	ùu.....CóóåÜ...
00403010	00 00 00 00 C0 3F CD CC CC 3F 33 33 F3 3F 33 33	....À?ÍÌÌ?33ó?33
00403020	F3 BF A1 32 7D 0E 00 00 00 00 01 01 01 01 01 01	ó¿¡2}..........
00403030	01 01 01 01 01 01 01 01 01 01 00 00 00 00 00 00	....

Адрес	Шестанадцатеричное	ASCII
00403000	F9 75 00 00 01 02 03 00 43 F3 F3 E5 DC 05 00 00	ùu.....CóóåÜ...
00403010	00 00 00 00 C0 3F CD CC CC 3F 33 33 F3 3F 33 33	....À?ÍÌÌ?33ó?33
00403020	F3 BF A1 32 7D 0E 00 00 00 00 01 01 01 01 01 01	ó¿¡2}..........
00403030	01 01 01 01 01 01 01 01 01 01 00 00 00 00 00 00	....

Название перемен-ной	Начальный адрес	Конечный адрес	Размер данных, байт	Описание
a	00403000	00403007	8	Массив из двух 4-байтовых чисел
b	00403008	0040300B	4	Массив из четырех 1-байтовых чисел
-	0040300C	00403011	6	Неименовая область, содержащая число размером 6 байт
-	00403012	00403021	16	Неименовая область, содержащая четыре вещественных числа длиной 4 байт
t	00403022	00403029	8	8-байтовое число
stra	0040302A	00403039	16	Массив из 16 1-байтовых чисел
Общий размер сегмента данных:			58	

5. Пошаговая трассировка программы

```
00401000 <lab1.OptionalHeade  BE 41737365          mov esi,65737341                    OptionalHeader.Ad
00401005                      2335 08304000        and esi,dword ptr ds:[403008]
0040100B                      8935 2A304000        mov dword ptr ds:[40302A],esi
00401011                      8B0D 22304000        mov ecx,dword ptr ds:[403022]
00401017                      6BC9 07              imul ecx,ecx,7
0040101A                      83C1 06              add ecx,6
0040101D                      890D 2E304000        mov dword ptr ds:[40302E],ecx
00401023                      8005 32304000 71     add byte ptr ds:[403032],71
0040102A                      FE0D 33304000        dec byte ptr ds:[403033]
00401030                      68 2A304000          push lab1.40302A
00401035                      FF15 08204000        call dword ptr ds:[<puts>]
0040103B                      83C4 04              add esp,4
0040103E                      FF15 0C204000        call dword ptr ds:[<_getch>]
00401044                      6A 00                push 0
00401046                      E8 01000000          call <JMP.&ExitProcess>
0040104B                      CC                   int3
0040104C <JMP.&ExitProcess>   FF25 00204000        jmp dword ptr ds:[<ExitProcess>]    JMP.&ExitProcess
00401052                      0000                 add byte ptr ds:[eax],al
00401054                      0000                 add byte ptr ds:[eax],al
00401056                      0000                 add byte ptr ds:[eax],al
```

Исходное состояние регистров:

```
EAX  7FFD1000
EBX  7FFD1000
ECX  00000000
EDX  00401000
EBP  0050FF68
ESP  0050FF54
ESI  00000000
EDI  00000000

EIP  00401000

EFLAGS  00000246
ZF 1  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1
```

```
BE 41737365          mov esi,65737341
```

```
EAX  7FFD1000
EBX  7FFD1000
ECX  00000000
EDX  00401000
EBP  0050FF68
ESP  0050FF54
ESI  65737341
EDI  00000000

EIP  00401005

EFLAGS  00000246
ZF 1  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1
```

ESI = 65737341h

```
2335 08304000        and esi,dword ptr ds:[403008]
```

```
EAX  7FFD1000
EBX  7FFD1000
ECX  00000000
EDX  00401000
EBP  0050FF68
ESP  0050FF54
ESI  65737341
EDI  00000000

EIP  0040100B

EFLAGS  00000206
ZF 0  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1
```

ESI = ESI & E5F3F343(т.к. DB просто определяет байты, а мы обращаемся к DD поэтому нужно развернуть)

```
mov dword ptr ds:[40302A],esi
```

```
EAX  7FFD1000
EBX  7FFD1000
ECX  00000000
EDX  00401000
EBP  0050FF68
ESP  0050FF54
ESI  65737341
EDI  00000000

EIP  00401011

EFLAGS  00000206
ZF 0  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1
```

В начало stra записали содержимое регистра ESI

```
mov ecx,dword ptr ds:[403022]
```

```
EAX   7FFD1000
EBX   7FFD1000
ECX   0E7D32A1
EDX   00401000
EBP   0050FF68
ESP   0050FF54
ESI   65737341
EDI   00000000

EIP   00401017

EFLAGS  00000206
ZF 0  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1
```

Записали в ECX значение t

```
imul ecx,ecx,7
```

```
EAX   7FFD1000
EBX   7FFD1000
ECX   656C6267
EDX   00401000
EBP   0050FF68
ESP   0050FF54
ESI   65737341
EDI   00000000

EIP   0040101A

EFLAGS  00000206
ZF 0  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1
```

ECX = ECX * 7 (656C6267h)

```
add ecx,6
```

```
EAX   7FFD1000
EBX   7FFD1000
ECX   656C626D
EDX   00401000
EBP   0050FF68
ESP   0050FF54
ESI   65737341
EDI   00000000

EIP   0040101D

EFLAGS  00000202
ZF 0  PF 0  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1
```

ECX = ECX + 6 (656C626D)

```
890D 2E304000    mov dword ptr ds:[40302E],ecx
```

```
EAX  7FFD1000
EBX  7FFD1000
ECX  656C626D
EDX  00401000
EBP  0050FF68
ESP  0050FF54
ESI  65737341
EDI  00000000

EIP  00401023

EFLAGS  00000202
ZF 0  PF 0  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1
```

stra[4:7] = ECX (6D626C65h)

```
8005 32304000 71    add byte ptr ds:[403032],71
```

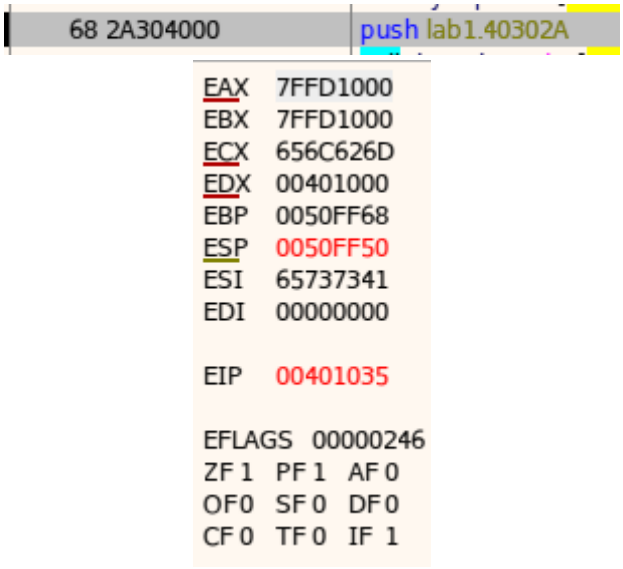
```
EAX  7FFD1000
EBX  7FFD1000
ECX  656C626D
EDX  00401000
EBP  0050FF68
ESP  0050FF54
ESI  65737341
EDI  00000000

EIP  0040102A

EFLAGS  00000206
ZF 0  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1
```

stra[8] = 71h (увеличили ячейку на 71)

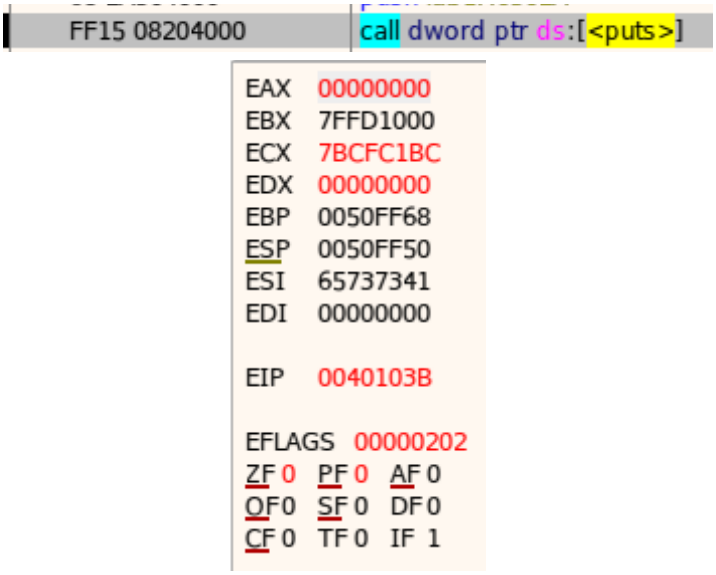
```
FE0D 33304000    dec byte ptr ds:[403033]
```

```
EAX  7FFD1000
EBX  7FFD1000
ECX  656C626D
EDX  00401000
EBP  0050FF68
ESP  0050FF54
ESI  65737341
EDI  00000000

EIP  00401030

EFLAGS  00000246
ZF 1  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1
```

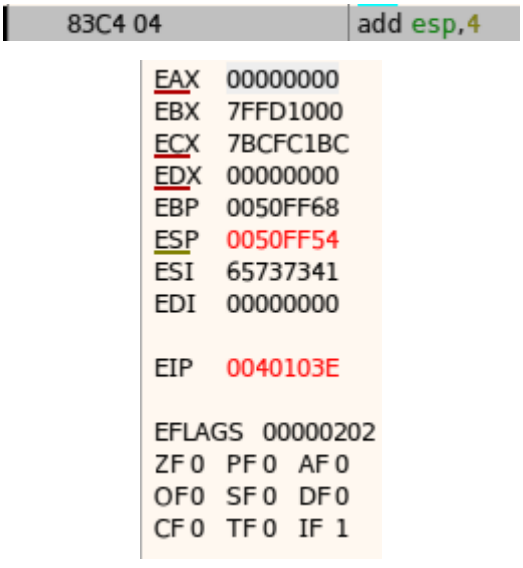
stra[9] -= 1 (0h)



Кладет в стек значение из ячейки с адресом 40302A



Вызов puts



Вызов puts

```
FF15 0C204000    call dword ptr ds:[<_getch>]
```

```
EAX  0000000D
EBX  7FFD1000
ECX  7BCFC1BC
EDX  00000000
EBP  0050FF68
ESP  0050FF54
ESI  65737341
EDI  00000000

EIP  00401044

EFLAGS  00000306
ZF 0  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 1  IF 1
```

Вызов getch

```
6A 00    push 0
```

```
EAX  0000000D
EBX  7FFD1000
ECX  7BCFC1BC
EDX  00000000
EBP  0050FF68
ESP  0050FF50
ESI  65737341
EDI  00000000

EIP  00401046

EFLAGS  00000206
ZF 0  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1
```

Кладет в стек 0

```
E8 01000000    call <JMP.&ExitProcess>
```

```
EAX  0000000D
EBX  7FFD1000
ECX  7BCFC1BC
EDX  00000000
EBP  0050FF68
ESP  0050FF50
ESI  65737341
EDI  00000000

EIP  00401046

EFLAGS  00000306
ZF 0  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 1  IF 1
```

Вызывает вызод из программы

Вывод: в ходе выполнения л.р. я получил навыки создания простейших ассемблерных программ с использованием пакета masm32 и научиться пользоваться отладчиком x32dbg.