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)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Лабораторная работа №1**

по дисциплине: «Архитектура вычислительных систем»

тема: «Разработка программ на ассемблере. Работа с отладчиком x32dbg, пакетом masm32»

**Выполнил студент группы ПВ-222**

Короткунов Александр Александрович

**Проверили**

Осипов Олег Васильевич

Белгород 2024 г.

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

1. Ознакомились со средой `x32dbg` и компилятором `masm32`.
2. Создали и скомпилировали программу в соответствии с вариантом задания. В программу включили комментарии с описанием, что делает каждая инструкция.

```
.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              ; массив из двух двойных слов
    b DB 43h, 0F3h, 0F3h, 0E5h      ; массив из четырёх слов
      DF 1500                       ; шестибайтовое число без имени
      DD 1.5, 1.6, 1.9, -1.9        ; массив из четырёх двойных слов без
имени
    t DQ 0E7D32A1h                  ; восьмибайтовое слово
    stra DB 16 dup(1)               ; массив слов с 16-ю единичками (1, 1,
1, 1...)

.code
start:
    MOV ESI, 65737341h              ; ESI = 65737341h
    AND ESI, dword ptr b            ; ESI = ESI & *b
    MOV dword ptr stra, ESI         ; stra[0-3] = ESI[0-3]
    MOV ECX, dword ptr t            ; ECX = *t
    IMUL ECX, 7                     ; ECX = ECX * 7
    ADD ECX, 6                      ; ECX = ECX + 6
    MOV dword ptr stra[4], ECX      ; stra[4-7] = ECX[0-3]
    ADD stra[8], 'q'                ; stra[8] += 'q'
    DEC stra[9]                     ; stra[9] -= 1

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

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

3. С помощью отладчика определили местонахождение переменных, строк и массивов в сегменте данных, а также их размер. Составили таблицу и подробное описание ячеек сегмента данных.

| Название переменной | Начальный адрес | Конечный адрес | Размер данных, байт | Описание |
|---|---|---|---|---|
| a | 00403000 | 00403007 | 8 | Массив из двух 4-байтовых целых чисел 30201 и 0x30201 |
| b | 00403008 | 0040300B | 4 | Массив из четырёх однобайтовых целых 0x43, 0xF3, 0xF3, 0xE5 |
| - | 0040300C | 00403011 | 6 | Неинициализированная 6-байтовая переменная целочисленного типа со значением 1500 |
| - | 00403012 | 00403021 | 16 | Неинициализированный массив из четырёх 4-байтовых вещественных переменных 1.5, 1.6, 1.9, -1.9 |
| t | 00403022 | 00403029 | 8 | 8-байтовое целое число 0x0E7D32A1 |
| stra | 0040302A | 00403039 | 16 | Массив 16-ти 1-байтовых целых чисел 1 |
| Общий размер сегмента данных: | | | 58 | |

Ячейки памяти с адресами от `0x00403000` до `0x00403007` содержат два 4-байтовых целых числа: `30201` и `30201h`. Первое число `30201` представлено в десятичной системе, а второе — в шестнадцатеричной. В памяти они представлены следующим образом:

```
F9 75 00 00 01 02 03 00
```

Далее, начиная с адреса `0x00403008` до `0x0040300B`, расположен массив из четырёх 1-байтовых целых чисел:

- `0x43` (67 в десятичной системе)
- `0xF3` (243 в десятичной системе)
- `0xF3` (243 в десятичной системе)
- `0xE5` (229 в десятичной системе)

Эти байты располагаются последовательно в памяти:

```
43 F3 F3 E5
```

Следующие ячейки памяти с адресами от `0x0040300C` до `0x00403011` содержат шестибайтовую переменную без имени с целочисленным значением `1500`. Число `1500` в шестнадцатеричной системе соответствует значению `0x5DC`. Это число будет храниться в памяти в виде:

```
DC 05 00 00 00 00
```

Ячейки памяти с адресами от `0x00403012` до `0x00403021` содержат массив из четырёх 4-байтовых вещественных чисел. Эти числа: `1.5`, `1.6`, `1.9` и `-1.9`, хранятся в формате IEEE 754. В памяти они будут представлены так:

- `1.5` в шестнадцатеричной форме равен `3FC00000`.
- `1.6` — `3FCCCCCD`.
- `1.9` — `3FF33333`.
- `-1.9` — `BFF33333`.

Представление этого массива в памяти будет следующим:

```
00 00 00 00 C0 3F CD CC CC 3F 33 33 F3 3F 33 33 F3 BF
```

Следующие ячейки памяти с адресами от `0x00403022` до `0x00403029` содержат 8-байтовое целое число `0x0E7D32A1`, которое хранится в памяти как:

```
A1 32 7D 0E 00 00 00 00
```

Начиная с адреса `0x0040302A` до `0x00403039` располагается массив `stra`, который содержит 16 однобайтовых целых чисел, все из которых инициализированы значением `1`. В памяти они выглядят как:

```
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
```

4. Выполнили пошаговую трассировку программы. Определили какие регистры, флаги и ячейки памяти изменяют свои значения в процессе выполнения команд. Обеспечили корректное завершение программы вызовом системной функции `ExitProcess` с кодом завершения `0`.

**Исходное состояние регистров:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| EAX= | 0019FFCC | EBX= | 00283000 | ECX= | 00401000 | EDX= | 00401000 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| EBP= | 0019FF84 | ESP= | 0019FF78 | ESI= | 00401000 | EDI= | 00401000 |
| EIP= | 00401000 | | | | | | |
| ZF= | 1 | PF= | 1 | AF= | 0 | | |
| OF= | 0 | SF= | 0 | DF= | 0 | | |
| CF= | 0 | TF= | 1 | IF= | 1 | | |

| mov esi, 0x65737341 | | | | КОП: | BE 41 73 73 65 | | |
|---|---|---|---|---|---|---|---|
| EAX= | 0019FFCC | EBX= | 00283000 | ECX= | 00401000 | EDX= | 00401000 |
| EBP= | 0019FF84 | ESP= | 0019FF78 | ESI= | 65737341 | EDI= | 00401000 |
| EIP= | 00401005 | | | | | | |
| ZF= | 1 | PF= | 1 | AF= | 0 | | |
| OF= | 0 | SF= | 0 | DF= | 0 | | |
| CF= | 0 | TF= | 0 | IF= | 1 | | |
| Перемещает значение 0x65737341 в регистр ESI. | | | | | | | |

| and esi,dword ptr ds:[403008] | | | | КОП: | 23 35 08 30 40 00 | | |
|---|---|---|---|---|---|---|---|
| EAX= | 0019FFCC | EBX= | 00283000 | ECX= | 00401000 | EDX= | 00401000 |
| EBP= | 0019FF84 | ESP= | 0019FF78 | ESI= | 65737341 | EDI= | 00401000 |
| EIP= | 0040100B | | | | | | |
| ZF= | 0 | PF= | 1 | AF= | 0 | | |
| OF= | 0 | SF= | 0 | DF= | 0 | | |
| CF= | 0 | TF= | 1 | IF= | 1 | | |
| Выполняет операцию "логическое и" между регистрами ESI и 4-байтовым целым, нахощимся по адресу 403008. По адресу 403008 находится массив из четырёх байт b. Устанавливает флаг ZF в значение 0. | | | | | | | |

| mov dword ptr ds:[40302A],esi | | | | КОП: | 89 35 2A 30 40 00 | | |
|---|---|---|---|---|---|---|---|
| EAX= | 0019FFCC | EBX= | 00283000 | ECX= | 00401000 | EDX= | 00401000 |
| EBP= | 0019FF84 | ESP= | 0019FF78 | ESI= | 65737341 | EDI= | 00401000 |
| EIP= | 00401011 | | | | | | |
| ZF= | 0 | PF= | 1 | AF= | 0 | | |
| OF= | 0 | SF= | 0 | DF= | 0 | | |
| CF= | 0 | TF= | 1 | IF= | 1 | | |
| Пересылает значение из регистра ESI в массив stra, который находится по адресу 40302A. Записывает первые 4 байта в данном массиве. | | | | | | | |

| mov ecx,dword ptr ds:[403022] | | | | КОП: | 8B 0D 22 30 40 00 | | |
|---|---|---|---|---|---|---|---|
| EAX= | 0019FFCC | EBX= | 00283000 | ECX= | 0E7D32A1 | EDX= | 00401000 |
| EBP= | 0019FF84 | ESP= | 0019FF78 | ESI= | 65737341 | EDI= | 00401000 |
| EIP= | 00401017 | | | | | | |
| ZF= | 0 | PF= | 1 | AF= | 0 | | |
| OF= | 0 | SF= | 0 | DF= | 0 | | |
| CF= | 0 | TF= | 1 | IF= | 1 | | |
| Записывает в ECX значение переменной t, которая находится по адресу 403022. | | | | | | | |

| imul ecx,ecx,7 | | | | КОП: | 6B C9 07 | | |
|---|---|---|---|---|---|---|---|
| EAX= | 0019FFCC | EBX= | 00283000 | ECX= | 656C6267 | EDX= | 00401000 |
| EBP= | 0019FF84 | ESP= | 0019FF78 | ESI= | 65737341 | EDI= | 00401000 |
| EIP= | 0040101A | | | | | | |
| ZF= | 0 | PF= | 0 | AF= | 0 | | |
| OF= | 0 | SF= | 0 | DF= | 0 | | |
| CF= | 0 | TF= | 1 | IF= | 1 | | |
| Выполняет умножение значения регистра ECX на число 7. Устанавливает значение флага PF в значение 0. | | | | | | | |

| add ecx,6 | | | | КОП: | 83 C1 06 | | |
|---|---|---|---|---|---|---|---|
| EAX= | 0019FFCC | EBX= | 00283000 | ECX= | 656C626D | EDX= | 00401000 |
| EBP= | 0019FF84 | ESP= | 0019FF78 | ESI= | 65737341 | EDI= | 00401000 |
| EIP= | 0040101D | | | | | | |
| ZF= | 0 | PF= | 0 | AF= | 0 | | |
| OF= | 0 | SF= | 0 | DF= | 0 | | |
| CF= | 0 | TF= | 1 | IF= | 1 | | |
| Складывает значение регистра ECX с 6 и записывает результат в ECX. | | | | | | | |

| mov dword ptr ds:[40302E],ecx | | | | КОП: | 89 0D 2E 30 40 00 | | |
|---|---|---|---|---|---|---|---|
| EAX= | 0019FFCC | EBX= | 00283000 | ECX= | 656C626D | EDX= | 00401000 |
| EBP= | 0019FF84 | ESP= | 0019FF78 | ESI= | 65737341 | EDI= | 00401000 |
| EIP= | 00401023 | | | | | | |
| ZF= | 0 | PF= | 0 | AF= | 0 | | |
| OF= | 0 | SF= | 0 | DF= | 0 | | |

| | | | | | |
|---|---|---|---|---|---|
| CF= | 0 | TF= | 1 | IF= | 1 |

Записывает значение ECX в массив stra, который находится по адресу 40302E, начиная с элемента с идексом 4.

| add byte ptr ds:[403032],71 | | | | КОП: | 80 05 32 30 40 00 71 | | |
|---|---|---|---|---|---|---|---|
| EAX= | 0019FFCC | EBX= | 00283000 | ECX= | 656C626D | EDX= | 00401000 |
| EBP= | 0019FF84 | ESP= | 0019FF78 | ESI= | 65737341 | EDI= | 00401000 |
| EIP= | 0040102A | | | | | | |
| ZF= | 0 | PF= | 1 | AF= | 0 | | |
| OF= | 0 | SF= | 0 | DF= | 0 | | |
| CF= | 0 | TF= | 1 | IF= | 1 | | |

Добавляет значение 71 ('q') в 8-ой по индексу элемент массива stra. Элемент находится по адресу 403032. Устанавливает значение флага PF в значение 1.

| dec byte ptr ds:[403033] | | | | КОП: | FE 0D 33 30 40 00 | | |
|---|---|---|---|---|---|---|---|
| EAX= | 0019FFCC | EBX= | 00283000 | ECX= | 656C626D | EDX= | 00401000 |
| EBP= | 0019FF84 | ESP= | 0019FF78 | ESI= | 65737341 | EDI= | 00401000 |
| EIP= | 00401030 | | | | | | |
| ZF= | 1 | PF= | 1 | AF= | 0 | | |
| OF= | 0 | SF= | 0 | DF= | 0 | | |
| CF= | 0 | TF= | 1 | IF= | 1 | | |

Уменьшает значение 9-го по индексу элемента массива stra на единицу. Элемент находится по адресу 403033. Устанавливает значение флага ZF в значение 1.

| push lab1.40302A | | | | КОП: | 68 2A 30 40 00 | | |
|---|---|---|---|---|---|---|---|
| EAX= | 0019FFCC | EBX= | 00283000 | ECX= | 656C626D | EDX= | 00401000 |
| EBP= | 0019FF84 | ESP= | 0019FF74 | ESI= | 65737341 | EDI= | 00401000 |
| EIP= | 00401035 | | | | | | |
| ZF= | 1 | PF= | 1 | AF= | 0 | | |
| OF= | 0 | SF= | 0 | DF= | 0 | | |
| CF= | 0 | TF= | 1 | IF= | 1 | | |

Помещает адрес массива stra в стек. Умешьшает значение регистра ESP на 4.

| call dword ptr ds:[<&puts>] | | | | КОП: | FF 15 08 20 40 00 | | |
|---|---|---|---|---|---|---|---|
| EAX= | 00000000 | EBX= | 00283000 | ECX= | C180921B | EDX= | 00000000 |

| EBP= | 0019FF84 | ESP= | 0019FF74 | ESI= | 65737341 | EDI= | 00401000 |
|---|---|---|---|---|---|---|---|
| EIP= | 0040103B | | | | | | |
| ZF= | 1 | PF= | 1 | AF= | 0 | | |
| OF= | 0 | SF= | 0 | DF= | 0 | | |
| CF= | 0 | TF= | 1 | IF= | 1 | | |
| Вызывает функцию ctr_puts, выводит значение массива stra в консоль. Устанавливает значение регистров EAX, ECX, EDX в значения 00000000, 8AB09B5A, 00000000 соответственно. | | | | | | | |

| add esp,4 | | | | КОП: | 83 C4 04 | | |
|---|---|---|---|---|---|---|---|
| EAX= | 00000000 | EBX= | 00283000 | ECX= | C180921B | EDX= | 00000000 |
| EBP= | 0019FF84 | ESP= | 0019FF78 | ESI= | 65737341 | EDI= | 00401000 |
| EIP= | 0040103E | | | | | | |
| ZF= | 0 | PF= | 1 | AF= | 0 | | |
| OF= | 0 | SF= | 0 | DF= | 0 | | |
| CF= | 0 | TF= | 1 | IF= | 1 | | |
| Увеличивает указатель стека на 4, тем самым очищая его. Устанавливает значение флага ZF в значение 0. | | | | | | | |

| call dword ptr ds:[<&_getch>] | | | | КОП: | FF 15 0C 20 40 00 | | |
|---|---|---|---|---|---|---|---|
| EAX= | 0000000D | EBX= | 00283000 | ECX= | C1809207 | EDX= | 0019FDD8 |
| EBP= | 0019FF84 | ESP= | 0019FF78 | ESI= | 65737341 | EDI= | 00401000 |
| EIP= | 00401044 | | | | | | |
| ZF= | 1 | PF= | 1 | AF= | 0 | | |
| OF= | 0 | SF= | 0 | DF= | 0 | | |
| CF= | 0 | TF= | 1 | IF= | 1 | | |
| Вызов функции ctr__getch. Ожидание ввода пользователя. Устанавливает значения регистров EAX, ECX, EDX в значения 0019FDD8, 8AB09B46 и 0019FDD8 соответственно. Устанавливает значение флага ZF в значение 1. | | | | | | | |

| push 0 | | | | КОП: | 6A 00 | | |
|---|---|---|---|---|---|---|---|
| EAX= | 0000000D | EBX= | 00283000 | ECX= | C1809207 | EDX= | 0019FDD8 |
| EBP= | 0019FF84 | ESP= | 0019FF74 | ESI= | 65737341 | EDI= | 00401000 |
| EIP= | 00401046 | | | | | | |
| ZF= | 1 | PF= | 1 | AF= | 0 | | |
| OF= | 0 | SF= | 0 | DF= | 0 | | |

| | | | | | |
|---|---|---|---|---|---|
| CF= | 0 | TF= | 1 | IF= | 1 |
| Добавляет число 0 в стек. 0 в данном случае - результат выполнения программы. То есть программа выполнилась без ошибок. Уменьшает значение регистра ESP на 4. | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| call <JMP.&ExitProcess> | | | | КОП: | E8 01 00 00 00 | | |
| EAX= | 0000000D | EBX= | 00283000 | ECX= | C1809207 | EDX= | 0019FDD8 |
| EBP= | 0019FF84 | ESP= | 0019FF70 | ESI= | 65737341 | EDI= | 00401000 |
| EIP= | 0040104C | | | | | | |
| ZF= | 1 | PF= | 1 | AF= | 0 | | |
| OF= | 0 | SF= | 0 | DF= | 0 | | |
| CF= | 0 | TF= | 1 | IF= | 1 | | |
| Вызов функции выхода из программы. | | | | | | | |

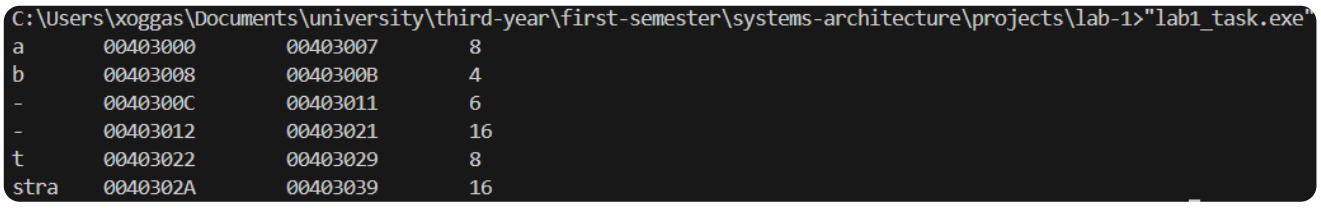
5. Необходимо вывести на ассемблере в консоль и в окно первые четыре столбца из первого задания.

```
.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)

    str_fmt DB "%s", 9, "%p", 9, "%p", 9, "%d", 13, 10, 0
    a_name DB "a", 0
    b_name DB "b", 0
    undefined DB "-", 0
    t_name DB "t", 0
    stra_name DB "stra", 0

.code
```

```
start:
    push offset b - offset a
    push offset a + (offset b - offset a) - 1
    push offset a
    push offset a_name

    push offset str_fmt
    call crt_printf
    add esp, 5

    push 4
    push offset b + 4 - 1
    push offset b
    push offset b_name

    push offset str_fmt
    call crt_printf
    add esp, 5

    push 6
    push offset b + 4 + 6 - 1
    push offset b + 4
    push offset undefined

    push offset str_fmt
    call crt_printf
    add esp, 5

    push 16
    push offset b + 4 + 6 + 16 - 1
    push offset b + 4 + 6
    push offset undefined

    push offset str_fmt
    call crt_printf
    add esp, 5

    push 8
    push offset t + 8 - 1
    push offset t
    push offset t_name

    push offset str_fmt
    call crt_printf
    add esp, 5

    push 16
    push offset stra + 16 - 1
    push offset stra
    push offset stra_name
```

```
    push offset str_fmt
    call crt_printf
    add esp, 5

    push 0
    call ExitProcess
END start
```

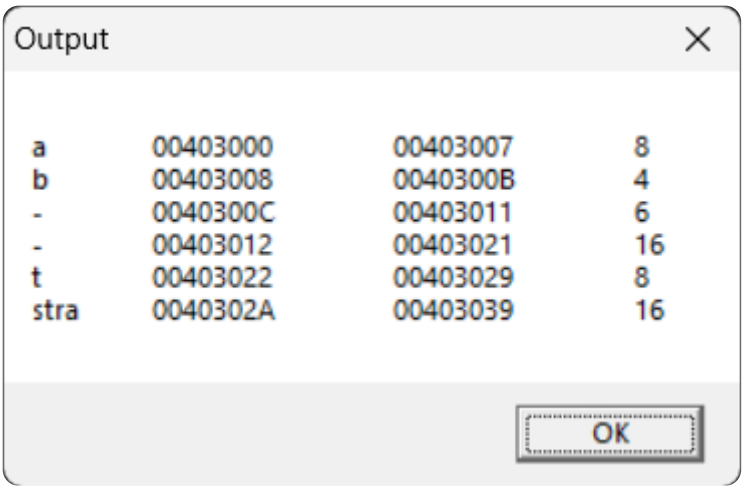
Результат выполнения программы:




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

include windows.inc
include kernel32.inc
include msvcrt.inc
include user32.inc
includelib kernel32.lib
includelib msvcrt.lib
includelib user32.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)

    str_fmt DB 6 dup("%s", 9, "%p", 9, "%p", 9, "%d", 10), 13, 0
    a_name DB "a", 0
    b_name DB "b", 0
    undefined DB "-", 0
    t_name DB "t", 0
    stra_name DB "stra", 0
    window_name DB "Output", 0
    buffer DB 74 dup(0)

.code
start:
    push 16
```

```
    push offset stra + 16 - 1
    push offset stra
    push offset stra_name

    push 8
    push offset t + 8 - 1
    push offset t
    push offset t_name

    push 16
    push offset b + 4 + 6 + 16 - 1
    push offset b + 4 + 6
    push offset undefined

    push 6
    push offset b + 4 + 6 - 1
    push offset b + 4
    push offset undefined

    push 4
    push offset b + 4 - 1
    push offset b
    push offset b_name

    push offset b - offset a
    push offset a + (offset b - offset a) - 1
    push offset a
    push offset a_name

    push offset str_fmt
    push offset buffer
    call crt_sprintf
    add esp, 6 * 4 + 2

    push 0
    push offset window_name
    push offset buffer
    push 0
    call MessageBoxA

    push 0
    call ExitProcess
end start
```

Результат выполнения программы:




**Вывод:** получили навыки создания простейших ассемблерных программ с использованием пакета `masm32` и научились пользоваться отладчиком `x32dbg`.