

Лабораторная работа №7

НКАбд-02-23

Выборнов Дмитрий Валерьевич

1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Задание

1. Реализация переходов в **NASM**.
2. Изучение структуры файлы листинга.
3. Задание для самостоятельной работы.

3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов: • условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия. • безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM.

Создаю каталог lab07 и файл lab7-1.asm.

Left	File	Command	Options	Right
<	~/work/arch-pc/lab07			.[^]>
.n	Name		Size	Modify time
/..			UP--DIR	Nov 25 15:19
lab7-1.asm			0	Nov 25 15:22

Рис. 4.1: Первый шаг.

Ввожу в файл lab7-1.asm текст нужной программы.

```
GNU nano 6.2
#include    'in_out.asm'

SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0

SECTION .text
GLOBAL _start
_start:

    jmp _label12

_label11:
    mov     eax, msg1
    call    sprintf

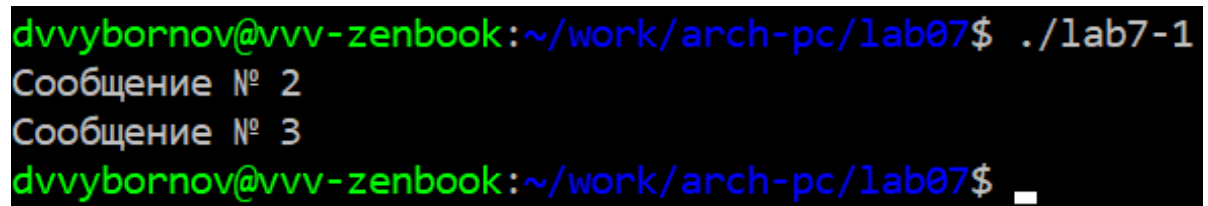
_label12:
    mov     eax, msg2
    call    sprintf

_label13:
    mov     eax, msg3
    call    sprintf

_end:
    call    quit_
```

Рис. 4.2: Второй шаг.

Создаю исполняемый файл и запускаю его.



```
dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$ _
```

Рис. 4.3: Третий шаг.

Изменяю текст программы.

```

#include    'in_out.asm'

SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0

SECTION .text
GLOBAL _start
_start:

    jmp _label2

_label1:
    mov     eax, msg1
    call    sprintf
    jmp     _end_

_label2:
    mov     eax, msg2
    call    sprintf
    jmp     _label1

_label3:
    mov     eax, msg3
    call    sprintf

_end:
    call    quit

```

Рис. 4.4: Четвёртый шаг.

Проверяю работу изменённого файла.

```
dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$ █
```

Рис. 4.5: Пятый шаг.

Изменяю работу файла, чтобы он выводил сообщения в обратном порядке.

```

#include    'in_out.asm'

SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0

SECTION .text
GLOBAL _start
_start:

    jmp _label3

_label1:
    mov     eax, msg1
    call    sprintfLF
    jmp     _end

_label2:
    mov     eax, msg2
    call    sprintfLF
    jmp     _label1

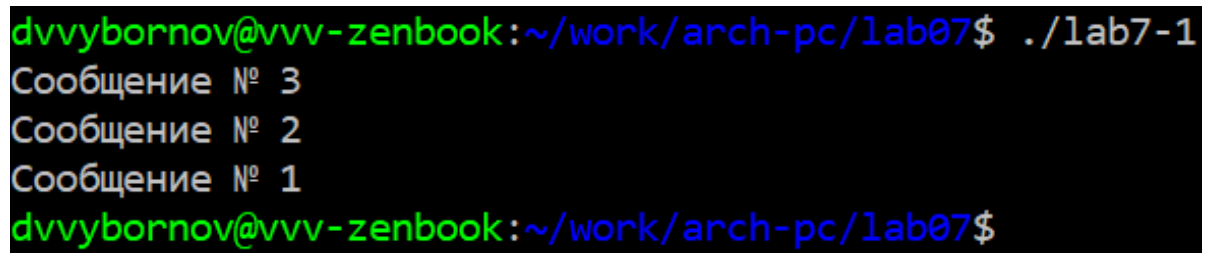
_label3:
    mov     eax, msg3
    call    sprintfLF
    jmp     _label2

_end:
    call    quit

```

Рис. 4.6: Шестой шаг.

Создаю исполняемый файл и запускаю его.

A terminal window with a black background and green text. The prompt is 'dvvybornov@vvv-zenbook:~/work/arch-pc/lab07\$'. The user enters './lab7-1'. The output consists of three lines: 'Сообщение № 3', 'Сообщение № 2', and 'Сообщение № 1'. The prompt appears again at the bottom.

```
dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$
```

Рис. 4.7: Седьмой шаг.

Создаю новый файл и ввожу в него текст программы, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных А, В и С.

```

GNU nano 6.2
#include 'in_out.asm'

SECTION .data
msg1 db 'Введите В: ', 0h
msg2 db "Наибольшее число: ", 0h
A dd '20'
C dd '50'
SECTION .bss
max resb 10
B resb 10
SECTION .text
GLOBAL _start
_start:
    mov eax, msg1
    call sprint

    mov ecx, B
    mov edx, 10
    call sread

    mov eax, B
    call atoi
    mov [B], eax

    mov ecx, [A]
    mov [max], ecx

    cmp ecx, [C]
    jg check_B
    mov ecx, [C]
    mov [max], ecx

    check_B:
    mov eax, max
    call atoi
    mov [max], eax

```

Рис. 4.8: Восьмой шаг.

Проверяю работу программы для нескольких значений В.

```
dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 40
Наибольшее число: 50
dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 999
Наибольшее число: 999
dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$
```

Рис. 4.9: Девятый шаг.

4.2 Изучение структуры файлы листинга.

Получаю файл листинга для lab7-2.asm.

```
dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Рис. 4.10: Десятый шаг.

Открывю файл листинга при помощи nano.

```

GNU nano 6.2 lab7-2.1st
1                               %include 'in_out.asm'
2                               <1> ;----- slen -----
3                               <1> ; Функция вычисления длины сообщения
4                               <1> slen:
5 00000000 53                   <1>     push    ebx
6 00000001 89C3                 <1>     mov     ebx, eax
7                               <1>
8                               <1> nextchar:
9 00000003 803800               <1>     cmp     byte [eax], 0
10 00000006 7403                 <1>     jz      finished
11 00000008 40                   <1>     inc     eax
12 00000009 EBF8                 <1>     jmp     nextchar
13                               <1>
14                               <1> finished:
15 0000000B 29D8                 <1>     sub     eax, ebx
16 0000000D 5B                   <1>     pop     ebx
17 0000000E C3                   <1>     ret
18                               <1>
19                               <1>
20                               <1> ;----- sprint -----
21                               <1> ; Функция печати сообщения
22                               <1> ; входные данные: mov eax,<message>
23                               <1> sprint:
24 0000000F 52                   <1>     push    edx
25 00000010 51                   <1>     push    ecx
26 00000011 53                   <1>     push    ebx
27 00000012 50                   <1>     push    eax
28 00000013 E8E8FFFFFF           <1>     call    slen
29                               <1>
30 00000018 89C2                 <1>     mov     edx, eax
31 0000001A 58                   <1>     pop     eax
32                               <1>
33 0000001B 89C1                 <1>     mov     ecx, eax
34 0000001D B801000000           <1>     mov     ebx, 1
35 00000022 B804000000           <1>     mov     eax, 4
36 00000027 CD80                 <1>     int     80h
37                               <1>

```

Рис. 4.11: Одиннадцатый шаг.

1. 15 - Номер строки “15”, адрес строки “0000000B”, машинный код “29D8”, исходный текст программы “sub eax, ebx” - sub уменьшает значение eax на значение ebx.
2. 2 - Пустая строка, разделяющая содержимое файлов in_out.asm и lab7-2.asm. (Технически является 172 строкой файла листинга.)
3. 19 - Номер строки “19”, адрес строки “000000FC”, машинный код “E842FFFFFF”, исходный текст программы “call sread” - call вызывает подпрограмму sread из файла in_out.asm. (Технически является 189 строкой листинга.)

Удаляю один из операндов инструкции mov.

```

_start:
    mov eax, msg1
    call sprint

    mov ecx, _
    mov edx, 10
    call sread

```

Рис. 4.12: Двенадцатый шаг.

При трансляции появилась ошибка, но, тем не менее, оба файла были созданы.

```

dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$ nasm -f elf -l lab7-3.lst lab7-3.asm
lab7-3.asm:17: error: invalid combination of opcode and operands
dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.lst lab7-2.o lab7-3.asm lab7-3.lst
dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$

```

Рис. 4.13: Тринадцатый шаг.

Также сообщение об ошибке было добавлено и в файл листинга.

```

17                                mov ecx,
17          *****              error: invalid combination of opcode and operands
18 000000F2 BA0A000000            mov edx, 10
19 000000F7 E847FFFFFF            call sread

```

Рис. 4.14: Четырнадцатый шаг.

4.3 Задание для самостоятельной работы.

4.3.1 №1

Создаю новый файл и ввожу в него текст программы, находящей наименьшее из чисел 41, 35 и 62.


```

GNU nano 6.2
#include 'in_out.asm'

SECTION .data
msg2 db 'Наименьшее число: ', 0h
A dd '41'
B dd '62'
C dd '35'
SECTION .bss
min resb 10
SECTION .text
GLOBAL _start
_start:

    mov ecx, [B]
    mov [min], ecx
    mov edx, [A]

    cmp ecx, ebx
    jl check_C
    mov ecx, [A]
    mov [min], ecx

    check_C:

    mov ecx, [min]
    cmp ecx, edx
    jl fin
    mov ecx, [C]
    mov [min], ecx

    mov eax, min
    call atoi
    mov [min], eax

    fin:
    mov eax, msg2
    call sprint

```

Рис. 4.15: Первый шаг первого задания.

Проверяю работу программы.

```
dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$ nano lab7-4.asm
dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$ ./lab7-4
Наименьшее число: 35
```

Рис. 4.16: Второй шаг первого задания.

4.3.2 №2

Создаю новый файл и ввожу в него текст программы, вычисляющей значение 10 варианта функции.

```

#include 'in_out.asm'

SECTION .data
msg1 db 'Type A ', 0h
msg2 db "Type b ", 0h
SECTION .bss
x resb 100
a resb 100
SECTION .text
GLOBAL _start
_start:
    mov ecx, x
    mov edx, 100
    call sread

    mov eax, x
    call atoi

    cmp eax, 2
    jge _steptwo

    mov ecx, a
    mov edx, 100
    call sread

    mov eax, a
    call atoi

    jmp _stepone

_stepone:

    mov ebx, 3
    mul ebx
    jmp _fin

_steptwo:

```

Рис. 4.17: Первый шаг второго задания.

Проверяю работу программы со значениями а и х, равными 3, 0 и 1, 2.

```
dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$ ./lab7-5
3
1
dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$ ./lab7-5
1
2
6
dvvybornov@vvv-zenbook:~/work/arch-pc/lab07$
```

Рис. 4.18: Второй шаг второго задания.

5 Выводы

Выполнив эту лабораторную работу, я изучил команды условного и безусловного переходов, приобрёл навыки написания программ с использованием переходов и познакомился с назначением и структурой файла листинга.