

**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7**

дисциплина: *Архитектура компьютера*

Студент: Довашеев Дмитрий

Группа: НКАбд-07-25

**МОСКВА**

2025 г.

# **Содержание**

<b>1 Цель работы</b>	<b>5</b>
<b>2 Задание</b>	<b>6</b>
<b>3 Теоретическое введение</b>	<b>7</b>
<b>4 Выполнение лабораторной работы</b>	<b>8</b>
4.1 Реализация переходов в NASM . . . . .	8
4.2 Изучение структуры файла листинга . . . . .	15
4.3 Задания для самостоятельной работы . . . . .	18
<b>5 Выводы</b>	<b>25</b>
<b>Список литературы</b>	<b>26</b>

# **Список иллюстраций**

4.1 Создание каталога и файла для программы . . . . .	8
4.2 Сохранение программы . . . . .	9
4.3 Запуск программы . . . . .	9
4.4 Изменение программы . . . . .	10
4.5 Запуск измененной программы . . . . .	11
4.6 Изменение программы . . . . .	12
4.7 Проверка изменений . . . . .	13
4.8 Сохранение новой программы . . . . .	14
4.9 Проверка программы из листинга . . . . .	15
4.10 Проверка файла листинга . . . . .	16
4.11 Удаление операнда из программы . . . . .	17
4.12 Просмотр ошибки в файле листинга . . . . .	17
4.13 Первая программа самостоятельной работы . . . . .	18
4.14 Проверка работы первой программы . . . . .	21
4.15 Вторая программа самостоятельной работы . . . . .	22
4.16 Проверка работы второй программы . . . . .	24

# **Список таблиц**

# **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 файлов листинга
3. Самостоятельное написание программ по материалам лабораторной работы

### **3 Теоретическое введение**

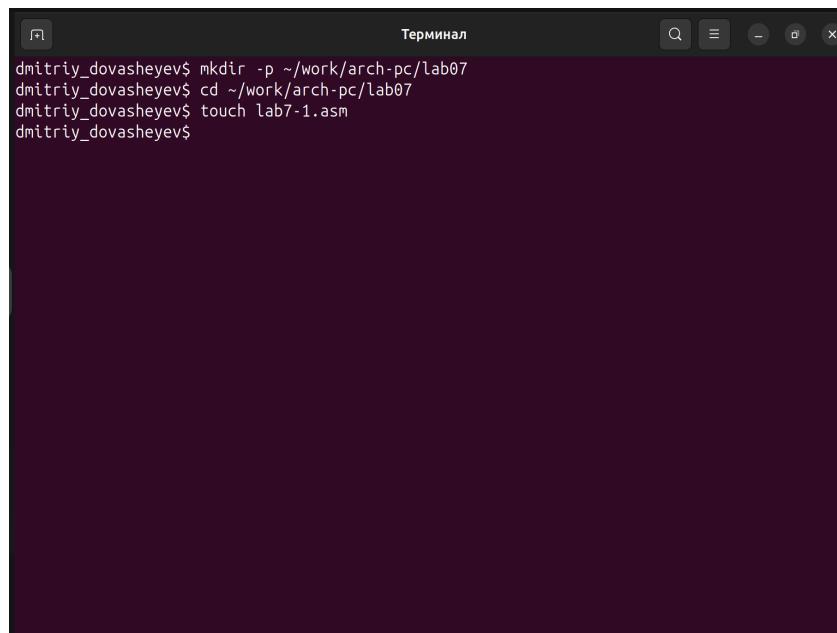
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- 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

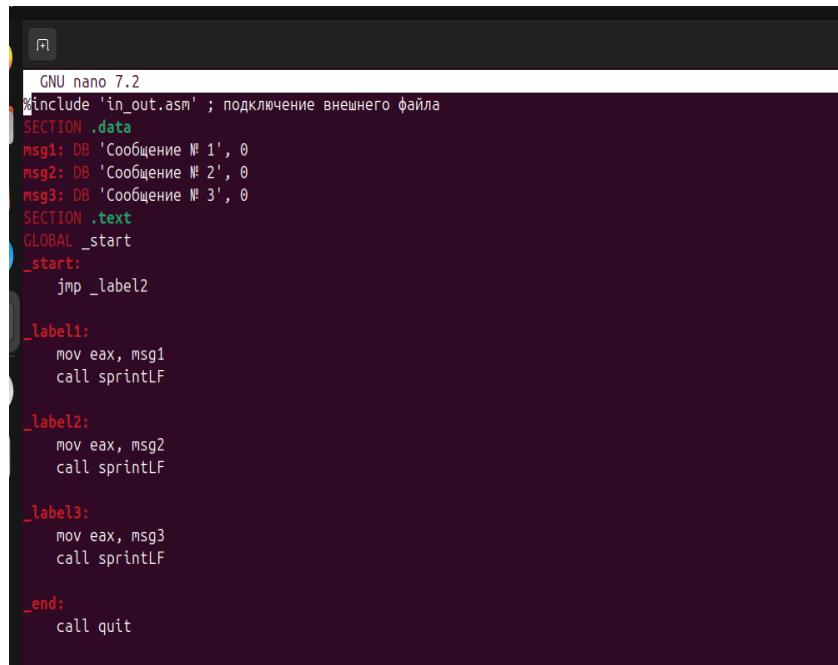
Создаю каталог для программ лабораторной работы №7 (рис. 4.1).



```
dmitriy_dovasheyev$ mkdir -p ~/work/arch-pc/lab07
dmitriy_dovasheyev$ cd ~/work/arch-pc/lab07
dmitriy_dovasheyev$ touch lab7-1.asm
dmitriy_dovasheyev$
```

Рис. 4.1: Создание каталога и файла для программы

Копирую код из листинга в файл будущей программы. (рис. 4.2).



```
GNU nano 7.2
#include 'in_out.asm' ; подключение внешнего файла
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 sprintLF

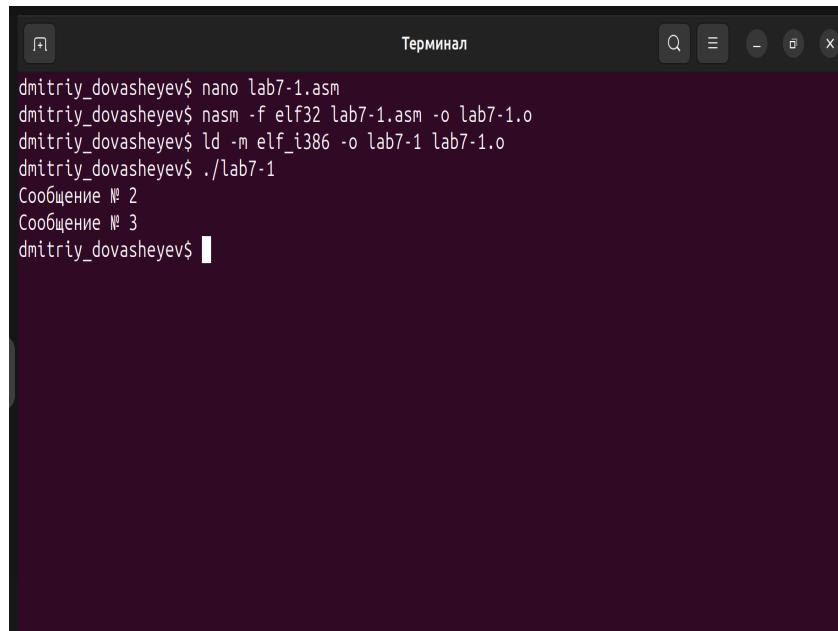
_label2:
    mov eax, msg2
    call sprintLF

_label3:
    mov eax, msg3
    call sprintLF

_end:
    call quit
```

Рис. 4.2: Сохранение программы

При запуске программы я убедился в том, что неусловный переход действительно изменяет порядок выполнения инструкций (рис. 4.3).

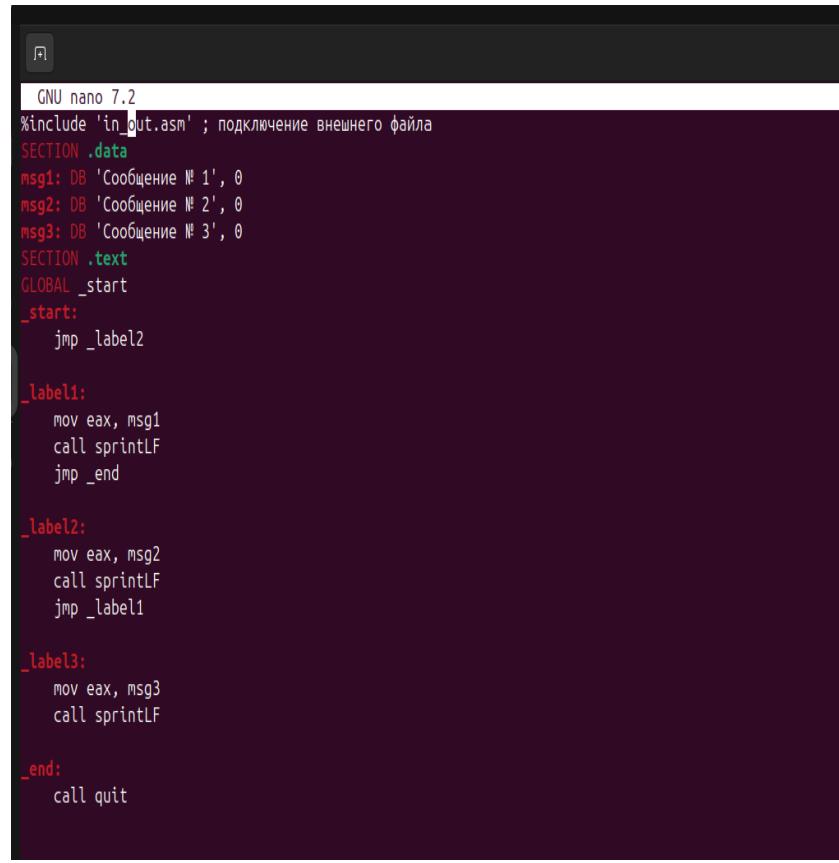


```
dmitriy_dovasheyev$ nano lab7-1.asm
dmitriy_dovasheyev$ nasm -f elf32 lab7-1.asm -o lab7-1.o
dmitriy_dovasheyev$ ld -m elf_i386 -o lab7-1 lab7-1.o
dmitriy_dovasheyev$ ./lab7-1
Сообщение № 2
Сообщение № 3
dmitriy_dovasheyev$
```

Рис. 4.3: Запуск программы

Изменяю программу таким образом, чтобы поменялся порядок выполнения

функций (рис. 4.4).



GNU nano 7.2

```
%include 'in_out.asm' ; подключение внешнего файла
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 sprintLF
    jmp _end

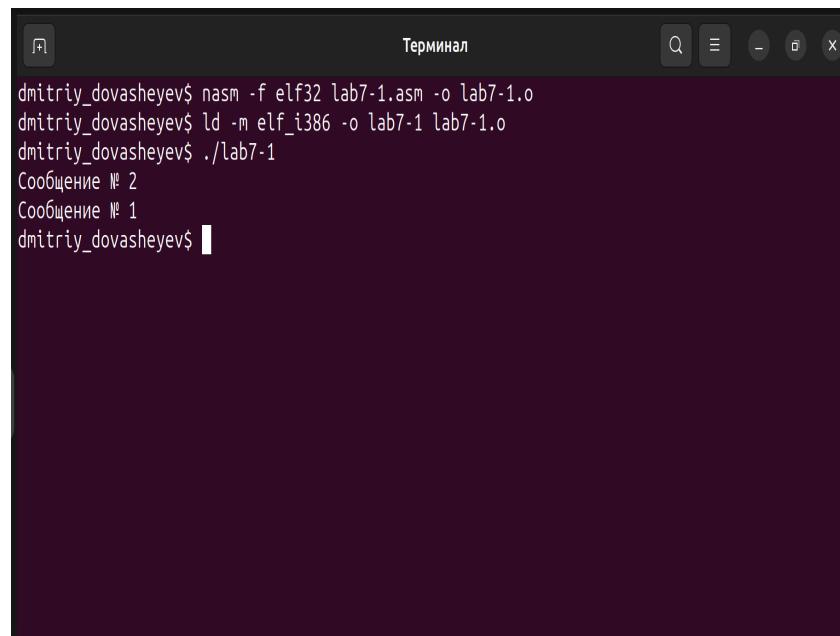
_label2:
    mov eax, msg2
    call sprintLF
    jmp _label1

_label3:
    mov eax, msg3
    call sprintLF

_end:
    call quit
```

Рис. 4.4: Изменение программы

Запускаю программу и проверяю, что примененные изменения верны (рис. 4.5).

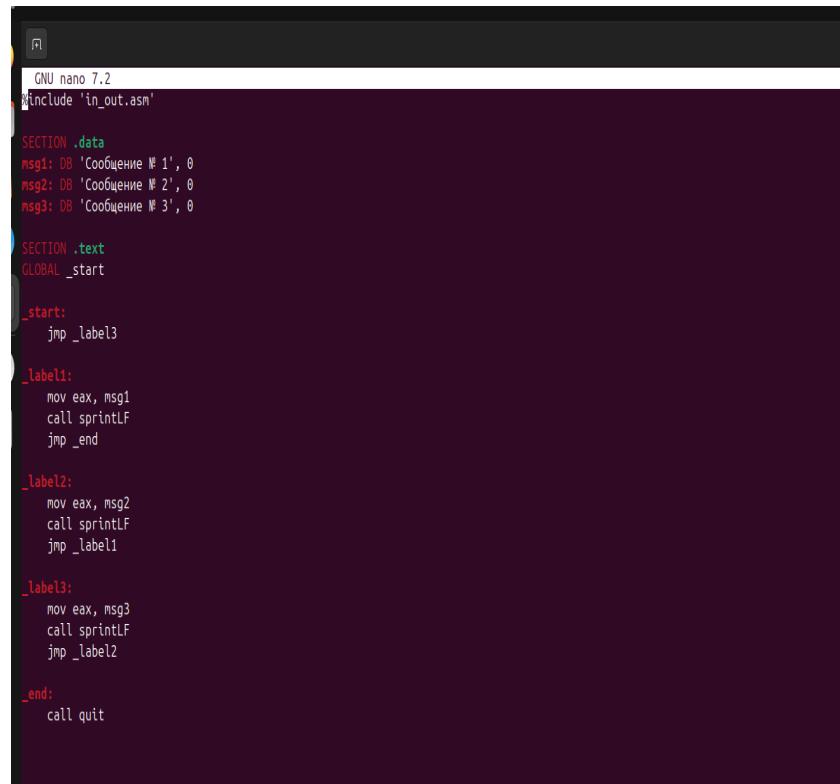


The screenshot shows a terminal window titled "Терминал". The terminal content is as follows:

```
dmitriy_dovasheyev$ nasm -f elf32 lab7-1.asm -o lab7-1.o
dmitriy_dovasheyev$ ld -m elf_i386 -o lab7-1 lab7-1.o
dmitriy_dovasheyev$ ./lab7-1
Сообщение № 2
Сообщение № 1
dmitriy_dovasheyev$
```

Рис. 4.5: Запуск измененой программы

Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. 4.6).



```
GNU nano 7.2
#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 _label3

_label1:
    mov eax, msg1
    call sprintLF
    jmp _end

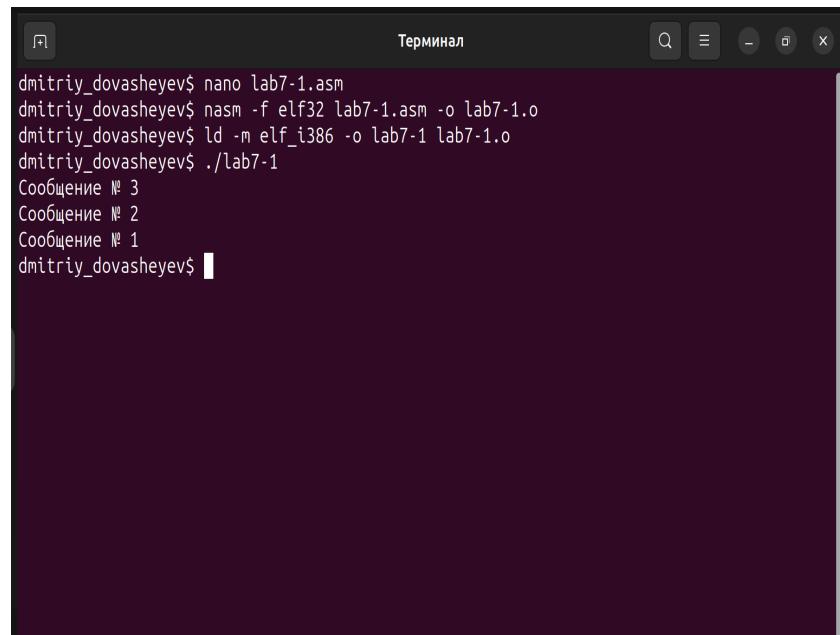
_label2:
    mov eax, msg2
    call sprintLF
    jmp _label1

_label3:
    mov eax, msg3
    call sprintLF
    jmp _label2

_end:
    call quit
```

Рис. 4.6: Изменение программы

Работа выполнена корректно, программа в нужном мне порядке выводит сообщения (рис. 4.7).



```
Терминал
dmitriy_dovasheyev$ nano lab7-1.asm
dmitriy_dovasheyev$ nasm -f elf32 lab7-1.asm -o lab7-1.o
dmitriy_dovasheyev$ ld -m elf_i386 -o lab7-1 lab7-1.o
dmitriy_dovasheyev$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
dmitriy_dovasheyev$
```

Рис. 4.7: Проверка изменений

Создаю новый рабочий файл и вставляю в него код из следующего листинга (рис. 4.8).

```
GNU nano 7.2
#include 'in_out.asm'

SECTION .data
msg1 db 'Введите B: ', 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

    mov ecx, [max]
    cmp ecx, [B]
    jg fin
    mov ecx, [B]
    mov [max], ecx

fin:
    mov eax, msg2
    call sprint

    mov eax, [max]
    call iprintLF

    call quit
```

Рис. 4.8: Сохранение новой программы

Программа выводит значение переменной с максимальным значением, проверяю работу программы с разными входными данными (рис. 4.9).

```
dmitriy_dovasheyev$ cd ~/work/arch-pc/lab07
dmitriy_dovasheyev$ touch lab7-2.asm
dmitriy_dovasheyev$ nano lab7-2.asm
dmitriy_dovasheyev$ nasm -f elf32 lab7-2.asm -o lab7-2.o
dmitriy_dovasheyev$ ld -m elf_i386 -o lab7-2 lab7-2.o
dmitriy_dovasheyev$ ./lab7-2
Введите В: 25
Наибольшее число: 50
dmitriy_dovasheyev$ ./lab7-2
Введите В: 60
Наибольшее число: 60
dmitriy_dovasheyev$ ./lab7-2
Введите В: 10
Наибольшее число: 50
dmitriy_dovasheyev$
```

Рис. 4.9: Проверка программы из листинга

## 4.2 Изучение структуры файла листинга

Создаю файл листинга с помощью флага `-l` команды `nasm` и открываю его с помощью текстового редактора `mousepad` (рис. 4.10).

```

Файл Правка Поиск Просмотр Документ Помощь
1   | 1  include <iostream.h>
2   | 2  int main()
3   | 3  {  

4 00000000 53C  | 4  pushad  

5 00000001 89C3 | 5  mov eax, esp  

6  | 6  nextchar:  

7   | 7  push eax  

8 00000003 803800 | 8  cmp byte [eax], 0  

9 00000006 7403 | 9  jne finished  

10 00000008 48 | 10  inc eax  

11 00000009 E8F8 | 11  jnc nextchar  

12  | 12  finished:  

13 0000000B 29D8 | 13  mov eax, ebx  

14 0000000E 58 | 14  pop ebx  

15 0000000F C3 | 15  ret  

16  | 16  message:  

17 00000000 48 | 17  pushad  

18  | 18  mov eax, message  

19 00000001 89C1 | 19  push eax  

20  | 20  sprint:  

21 0000000F 52 | 21  push edx  

22 00000010 51 | 22  push ecx  

23 00000011 53 | 23  push ebx  

24 00000012 58 | 24  push eax  

25 00000013 E8E8FFFF | 25  call Slen  

26  | 26  mov eax, ebx  

27 00000014 C3 | 27  pop ebx  

28 00000015 58 | 28  pop eax  

29 00000016 5A | 29  pop edx  

30 00000017 5A | 30  pop ecx  

31 00000018 5A | 31  pop ebx  

32 00000019 89C1 | 32  mov eax, ecx  

33 0000001D B801000000 | 33  mov ebx, 1  

34 0000001E 48 | 34  mov eax, 4  

35 00000027 CD88 | 35  int 80h  

36  | 36  message2:  

37 00000029 5B | 37  pop ebx  

38 0000002A 5A | 38  pop eax  

39 0000002B 5A | 39  pop edx  

40 0000002C C3 | 40  ret  

41  | 41  message3:  

42 0000002D 48 | 42  pushad  

43  | 43  sprintLF:  

44 0000002E 48 | 44  pushad  

45 0000002F 50 | 45  call sprint  

46 00000030 E800FFFFFF | 46  jmp _main  

47 00000032 50 | 47  push eax  

48 00000033 B80A000000 | 48  mov eax, 0AH  

49 00000034 48 | 49  push ebx  

50 00000035 89E8 | 50  mov eax, esp  

51 00000036 E8CFFFFFFF | 51  call sprint  

52 00000037 58 | 52  pop ebx  

53 00000041 58 | 53  pop eax  

54 00000042 C3 | 54  ret  

55  | 55  message4:  

56 00000043 5A | 56  pushad  

57 00000044 50 | 57  push eax  

58 00000045 B800000000 | 58  mov ebx, 0  

59 00000046 B803000000 | 59  mov eax, 3  

60 00000047 CD88 | 60  int 80h  

61  | 61  message5:  

62 00000048 5B | 62  pop ebx  

63 00000049 5A | 63  pop eax  

64 00000050 5A | 64  pop edx  

65 00000051 5A | 65  pop ecx  

66 00000052 59 | 66  pop ebx  

67 00000053 C3 | 67  ret  

68  | 68  message6:  

69 00000054 48 | 69  pushad  

70 00000055 51 | 70  push eax  

71 00000056 52 | 71  push edx  

72 00000057 50 | 72  push ecx  

73 00000058 48 | 73  pushad  

74 00000059 48 | 74  call printf  

75  | 75  message7:  

76 0000005A 48 | 76  pushad  

77 0000005B 50 | 77  push eax  

78 0000005C 51 | 78  push edx  

79 0000005D 52 | 79  push ecx  

80 0000005E 50 | 80  push ebx  

81 0000005F 48 | 81  pushad

```

Рис. 4.10: Проверка файла листинга

Первое значение в файле листинга - номер строки, и он может вовсе не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно идет сам машинный код, а заключает строку исходный текст программы с комментариями.

Удаляю один operand из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. 4.11).

```

GNU nano 7.2
%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
    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
    mov ecx, [max]
    cmp ecx, [B]
    jg fin
    mov ecx, [B]
    mov [max], ecx
fin:
    mov eax, msg2
    call sprint
    mov eax, [max]
    call iprintf
    call quit

```

Рис. 4.11: Удаление операнда из программы

В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются. (рис. 4.12).

```

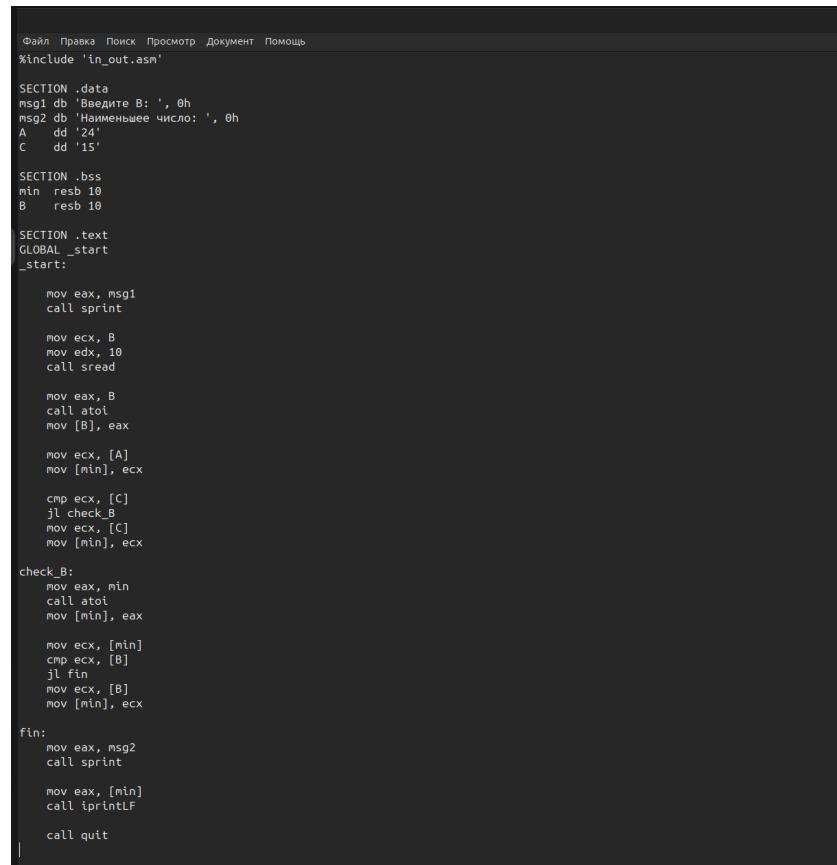
~ /work/arch-pc/lab07/lab7-2.lst - Mousepad
Файл Правка Поиск Просмотр Документ Помощь
17 000000E8 B8[00000000]      mov eax, msg1
18 000000ED E81DFFFFFF        call sprint
19
20 000000F2 B9[0A000000]      mov ecx, B
21 000000F7 BA0A000000        mov edx, 10
22 000000FC E842FFFFFF        call sread
23
24          *****
24          ***** error: invalid combination of opcode and operand
25 00000101 E896FFFFFF        call atoi
26 00000106 A3[0A000000]      mov [B], eax
27
28 0000010B 8B0D[35000000]      mov ecx, [A]
29 00000111 890D[00000000]      mov [max], ecx
30
31 00000117 3B0D[39000000]      cmp ecx, [C]
32 0000011D 7F0C                jg check_B
33 0000011F 8B0D[39000000]      mov ecx, [C]
34 00000125 890D[00000000]      mov [max], ecx
35
36          check_B:
37 0000012B B8[00000000]      mov eax, max
38 00000130 E867FFFFFF        call atoi
39 00000135 A3[00000000]      mov [max], eax

```

Рис. 4.12: Просмотр ошибки в файле листинга

## 4.3 Задания для самостоятельной работы

В рамках самостоятельной работы реализую программу поиска наименьшего из трех чисел А, В и С по выбранному варианту. Для выполнения задания использую девятый вариант, параметры которого заданы в методических указаниях. Модифицирую программу таким образом, чтобы она выводила переменную с минимальным значением (рис. 4.13).



```
Файл Правка Поиск Просмотр Документ Помощь
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите В: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '24'
C dd '15'

SECTION .bss
min 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 [min], ecx

    cmp ecx, []
    jl check_B
    mov ecx, []
    mov [min], ecx

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

    mov ecx, [min]
    cmp ecx, [B]
    jl fin
    mov ecx, [B]
    mov [min], ecx

fin:
    mov eax, msg2
    call sprint

    mov eax, [min]
    call iprintf
    call quit
```

Рис. 4.13: Первая программа самостоятельной работы

Код первой программы:

```
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите В: ', 0h
```

```
msg2 db 'Наименьшее число: ', 0h
A dd '24'
C dd '15'
```

```
SECTION .bss
min 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 [min], ecx
```

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

```
mov [min], ecx
```

```
check_B:
```

```
    mov eax, min
```

```
    call atoi
```

```
    mov [min], eax
```

```
    mov ecx, [min]
```

```
    cmp ecx, [B]
```

```
    jb fin
```

```
    mov ecx, [B]
```

```
    mov [min], ecx
```

```
fin:
```

```
    mov eax, msg2
```

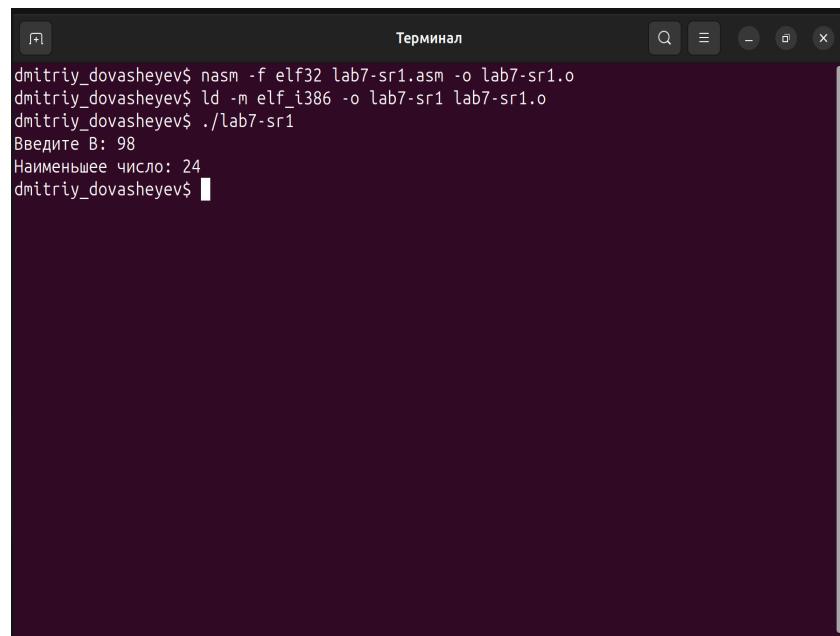
```
    call sprint
```

```
    mov eax, [min]
```

```
    call iprintLF
```

```
    call quit
```

Проверяю корректность написания первой программы (рис. 4.14).



The screenshot shows a terminal window titled "Терминал". The command line shows the user's path: dmitriy\_dovasheyev\$. The user runs three commands: nasm -f elf32 lab7-sr1.asm -o lab7-sr1.o, ld -m elf\_i386 -o lab7-sr1 lab7-sr1.o, and ./lab7-sr1. The output of the ./lab7-sr1 command is: Введите В: 98 Наименьшее число: 24. The terminal window has a dark background and light-colored text.

Рис. 4.14: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатурных переменных a и x (рис. 4.15).

```
Файл Правка Поиск Просмотр Документ Помощь
%include 'in_out.asm'

SECTION .data
msg_x db 'Введите значение переменной x: ', 0
msg_a db 'Введите значение переменной a: ', 0
rem db 'Результат: ', 0

SECTION .bss
x RESB 80
a RESB 80

SECTION .text
GLOBAL _start
_start:

    mov eax, msg_x
    call sprint
    mov ecx, x
    mov edx, 80
    call sread
    mov eax, x
    call atoi
    mov edi, eax      ; x -> EDI

    mov eax, msg_a
    call sprint
    mov ecx, a
    mov edx, 80
    call sread
    mov eax, a
    call atoi
    mov esi, eax      ; a -> ESI

    cmp edi, esi
    jle add_values
    mov eax, esi
    jmp print_result

add_values:
    mov eax, edi
    add eax, esi

print_result:
    mov edi, eax
    mov eax, rem
    call sprint
    mov eax, edi
    call tprintf
    call quit
|
```

Рис. 4.15: Вторая программа самостоятельной работы

Код второй программы:

```
%include 'in_out.asm'

SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
res: DB 'Результат: ', 0

SECTION .bss
x: RESB 80
a: RESB 80

SECTION .text
GLOBAL _start
_start:
    mov eax, msg_x
    call sprint
```

```
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax

mov eax, msg_a
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax

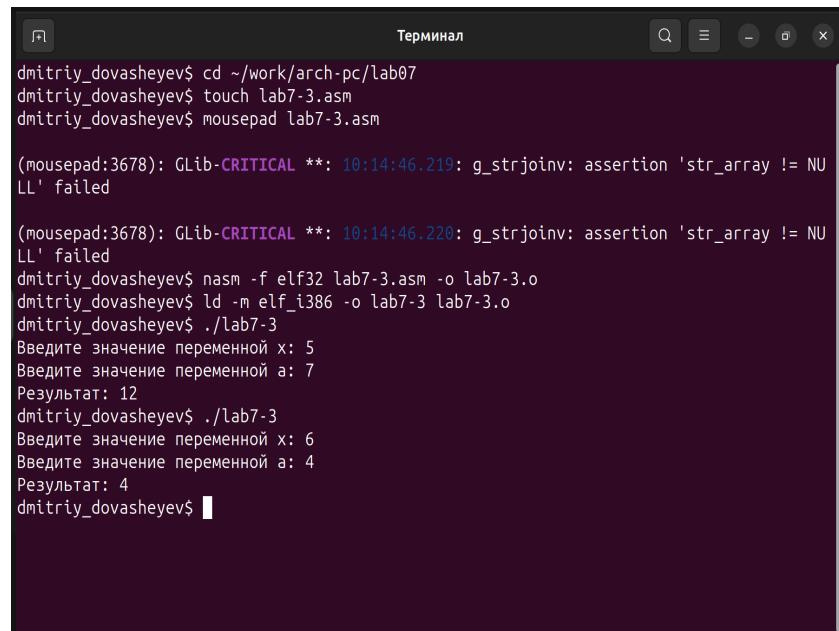
cmp edi, esi
jle add_values
mov eax, esi
jmp print_result

add_values:
mov eax, edi
add eax, esi

print_result:
mov edi, eax
mov eax, res
call sprint
```

```
mov eax, edi
call iprintLF
call quit
```

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений а и х (рис. 4.16).



```
Терминал
dmitriy_dovasheyev$ cd ~/work/arch-pc/lab07
dmitriy_dovasheyev$ touch lab7-3.asm
dmitriy_dovasheyev$ mousepad lab7-3.asm

(mousepad:3678): GLib-CRITICAL **: 10:14:46.219: g_strjoinv: assertion 'str_array != NULL' failed

(mousepad:3678): GLib-CRITICAL **: 10:14:46.220: g_strjoinv: assertion 'str_array != NULL' failed
dmitriy_dovasheyev$ nasm -f elf32 lab7-3.asm -o lab7-3.o
dmitriy_dovasheyev$ ld -m elf_i386 -o lab7-3 lab7-3.o
dmitriy_dovasheyev$ ./lab7-3
Введите значение переменной x: 5
Введите значение переменной a: 7
Результат: 12
dmitriy_dovasheyev$ ./lab7-3
Введите значение переменной x: 6
Введите значение переменной a: 4
Результат: 4
dmitriy_dovasheyev$
```

Рис. 4.16: Проверка работы второй программы

## **5 Выводы**

При выполнении лабораторной работы я изучил команды условных и безусловных переходов, а также приобрел навыки написания программ с использованием переходов, познакомился с назначением и структурой файлов листинга.

# **Список литературы**

1. Курс на ТУИС
2. Лабораторная работа №7
3. Программирование на языке ассемблера NASM Столяров А. В.