

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

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

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

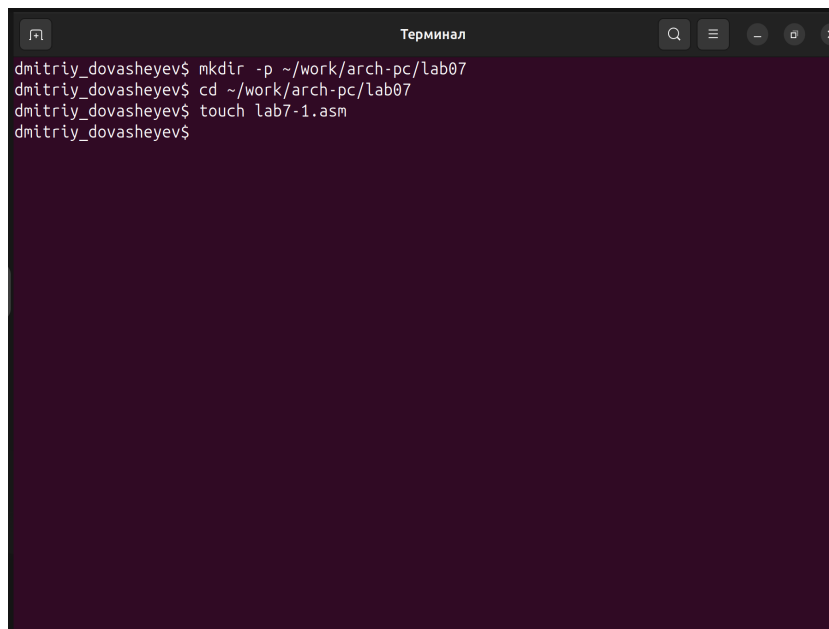
3 Теоретическое введение

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

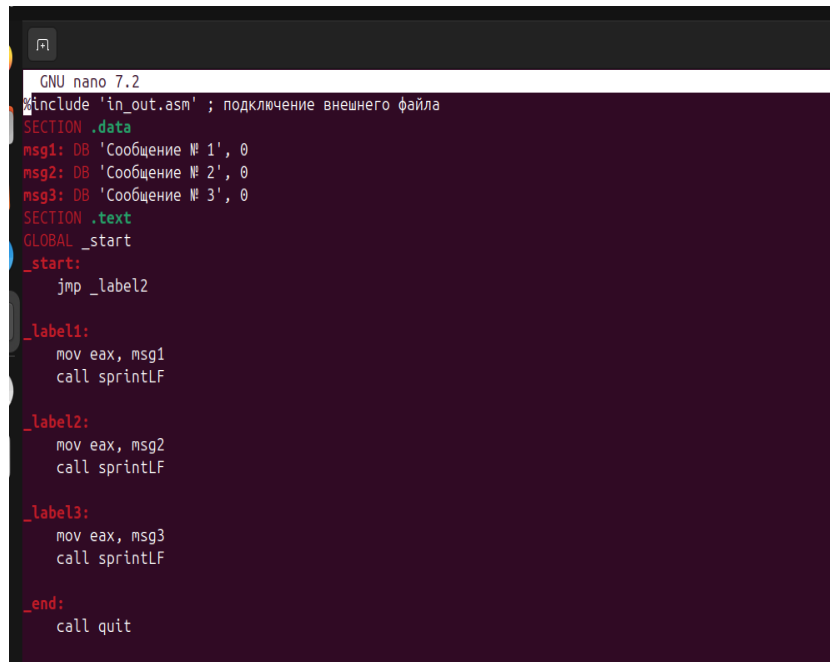
Создаю каталог для программ лабораторной работы №7 (рис. 4.1).

A screenshot of a terminal window titled "Терминал". The terminal shows a series of commands being executed by a user named "dmitriy_dovasheyev". The commands are: "mkdir -p ~/work/arch-pc/lab07", "cd ~/work/arch-pc/lab07", and "touch lab7-1.asm". The terminal background is dark purple, and the text is white. The window has standard Linux window controls (minimize, maximize, close) and a search icon in the top right corner.

```
Терминал
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

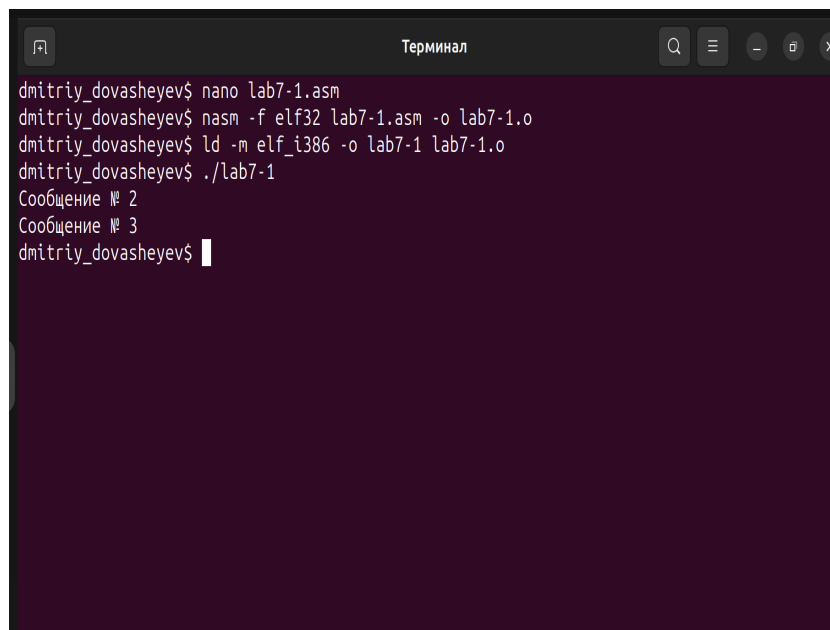
_label2:
    mov eax, msg2
    call sprintLF

_label3:
    mov eax, msg3
    call sprintLF

_end:
    call quit
```

Рис. 4.2: Сохранение программы

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

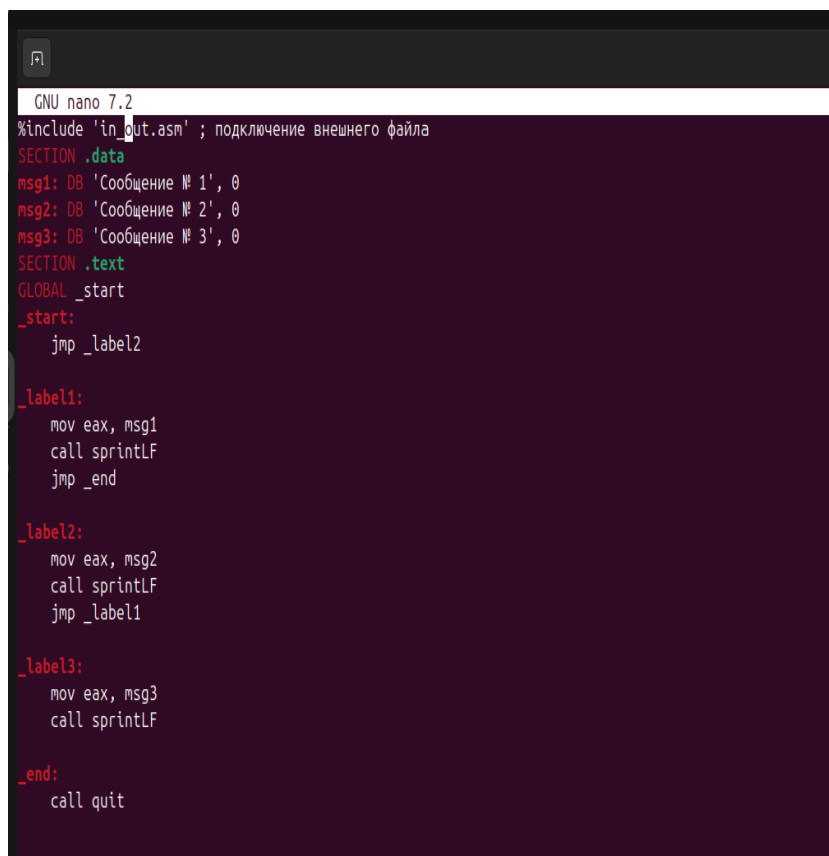


```
Терминал
dmitriy_dovasheyev$ nano lab7-1.asm
dmitriy_dovasheyev$ nasm -f elf32 lab7-1.asm -o lab7-1.o
dmitriy_dovasheyev$ ld -m elf_i386 -o lab7-1 lab7-1.o
dmitriy_dovasheyev$ ./lab7-1
Сообщение № 2
Сообщение № 3
dmitriy_dovasheyev$
```

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 sprintf
    jmp _end

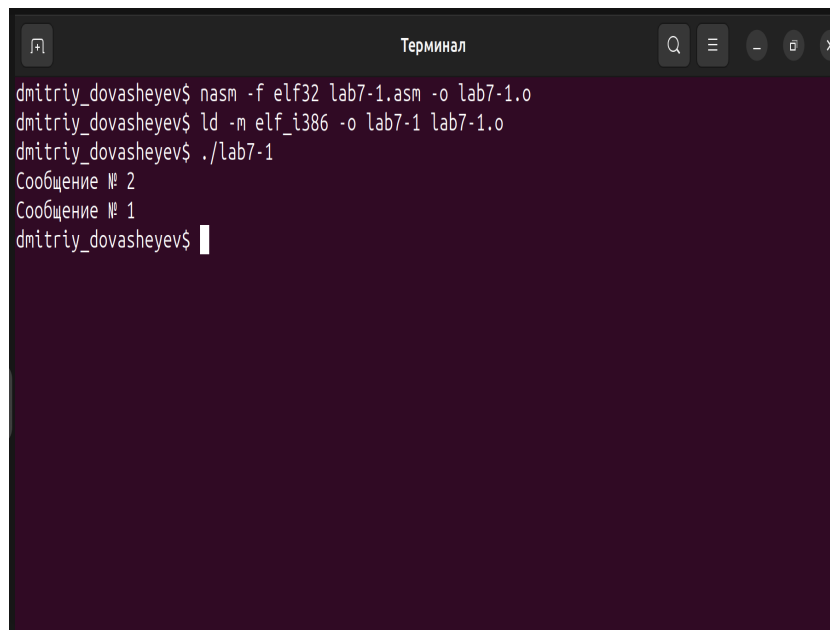
_label2:
    mov eax, msg2
    call sprintf
    jmp _label1

_label3:
    mov eax, msg3
    call sprintf

_end:
    call quit
```

Рис. 4.4: Изменение программы

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

A terminal window titled "Терминал" with standard window controls. It shows a series of commands and their outputs. The commands are: `nasm -f elf32 lab7-1.asm -o lab7-1.o`, `ld -m elf_i386 -o lab7-1 lab7-1.o`, and `./lab7-1`. The outputs are: "Сообщение № 2", "Сообщение № 1", and a prompt `dmitriy_dovasheyev$` with a cursor.

```
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 измененной 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 sprintf
    jmp _end

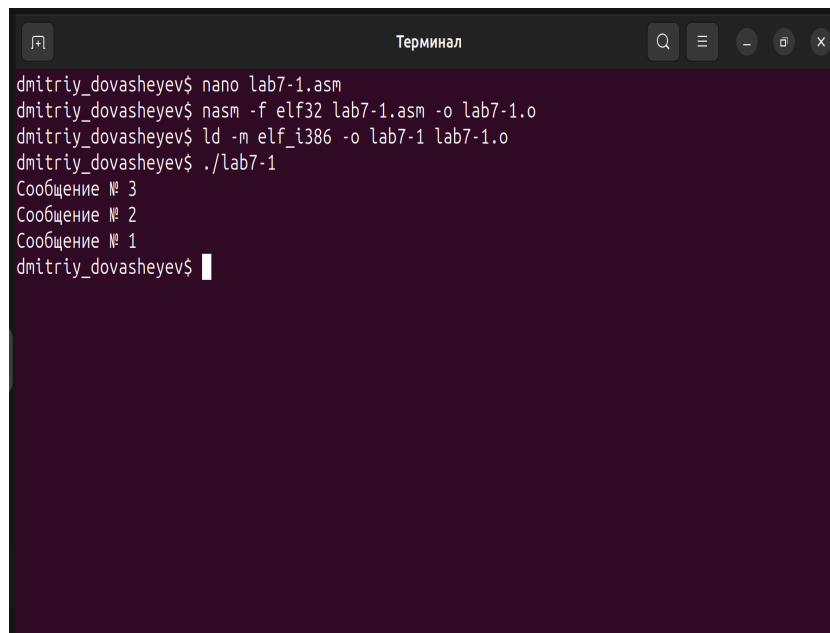
_label2:
    mov eax, msg2
    call sprintf
    jmp _label1

_label3:
    mov eax, msg3
    call sprintf
    jmp _label2

_end:
    call quit
```

Рис. 4.6: Изменение программы

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

A terminal window titled "Терминал" with a dark background. It shows a series of commands and their outputs. The user runs 'nano lab7-1.asm', then 'nasm -f elf32 lab7-1.asm -o lab7-1.o', then 'ld -m elf_i386 -o lab7-1 lab7-1.o', and finally './lab7-1'. The output shows three messages in Russian: "Сообщение № 3", "Сообщение № 2", and "Сообщение № 1".

```
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

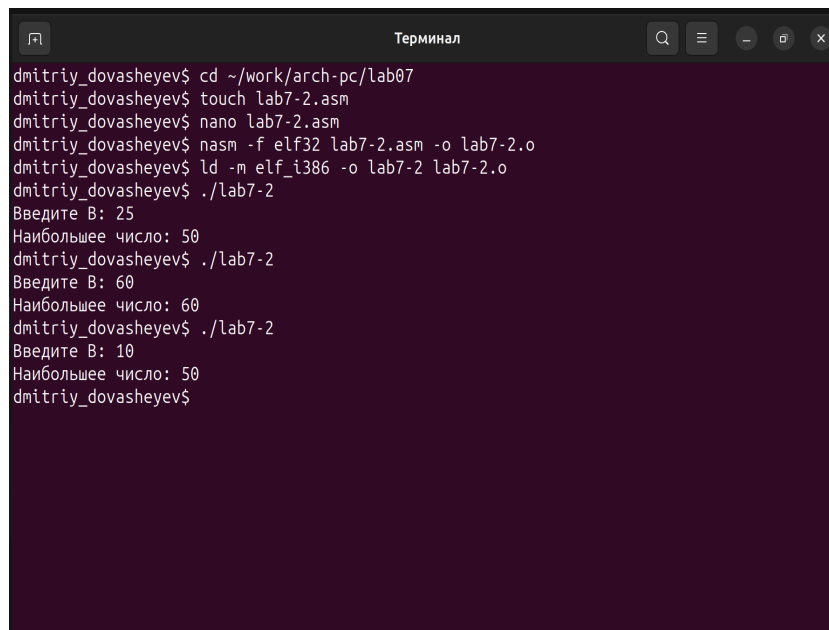
fin:
    mov eax, msg2
    call sprint

    mov eax, [max]
    call iprintfLF

    call quit
```

Рис. 4.8: Сохранение новой программы

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



```
Терминал
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 B: 25
Наибольшее число: 50
dmitriy_dovasheyev$ ./lab7-2
Введите B: 60
Наибольшее число: 60
dmitriy_dovasheyev$ ./lab7-2
Введите B: 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).

```

1 1
2 2
3 3
4 00000000 53
5 00000001 89C3
6 6
7 7
8 00000003 003800
9 00000005 7403
10 00000008 40
11 00000009 E7F8
12 12
13 13
14 00000008 2008
15 0000000D 5B
16 0000000E C3
17 17
18 18
19 19
20 20
21 21
22 22
23 0000000F 52
24 00000010 51
25 00000011 53
26 00000012 50
27 00000013 E8BFFFFFFF
28 28
29 0000001B 89C2
30 00000015 5B
31 31
32 0000001B 89C1
33 0000001D 8B01000000
34 00000027 8B04000000
35 00000027 CD00
36 36
37 00000025 5B
38 0000002A 59
39 00000026 54
40 0000002C C3
41 41
42 42
43 43
44 44
45 45
46 46
47 0000002D E8DFFFFFFF
48 48
49 00000032 59
50 00000033 8B0A000000
51 00000038 59
52 00000039 89C9
53 0000003B E8FFFFFFF
54 00000040 5B
55 00000041 5B
56 00000042 C3
57 57
58 58
59 59
60 60
61 61
62 00000043 52
63 00000044 5B
64 64
65 00000045 8B00000000
66 00000046 8B03000000
67 0000004F CD00
68 68
69 00000051 5B
70 00000052 59
71 00000053 C3
72 72
73 73
74 74
75 75
76 76
77 00000054 5B
78 00000055 51
79 00000056 52
80 00000057 56
81 00000058 00000000

```

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 операнд 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 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
    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 ftn
    mov ecx, [B]
    mov [max], ecx

ftn:
    mov eax, msg2
    call sprint

    mov eax, [max]
    call iprintLF

    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).

```

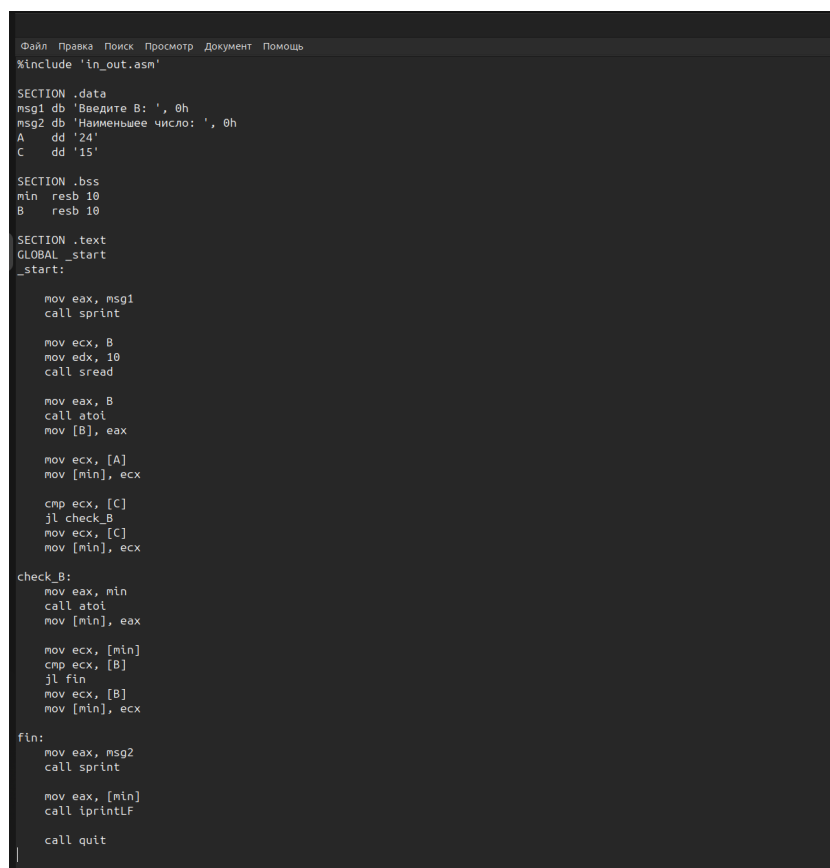
~/work/arch-pc/lab07/lab7-2.lst - Mousepad
Файл Правка Поиск Просмотр Документ Помощь
17 000000E8 B8[00000000]      mov eax, msg1
18 000000ED E81DFFFFFF      call sprint
19
20 000000F2 B9[0A000000]      mov ecx, B
21 000000F7 BA0A000000      mov edx, 10
22 000000FC E842FFFFFF      call sread
23
24                               mov eax
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 A, B и C 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 B: ', 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 sprintf

    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]
    jl check_B
    mov ecx, [C]
    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 sprintf

    mov eax, [min]
    call iprintfLF

    call quit
```

Рис. 4.13: Первая программа самостоятельной работы

Код первой программы:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg1 db 'Введите B: ', 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
```

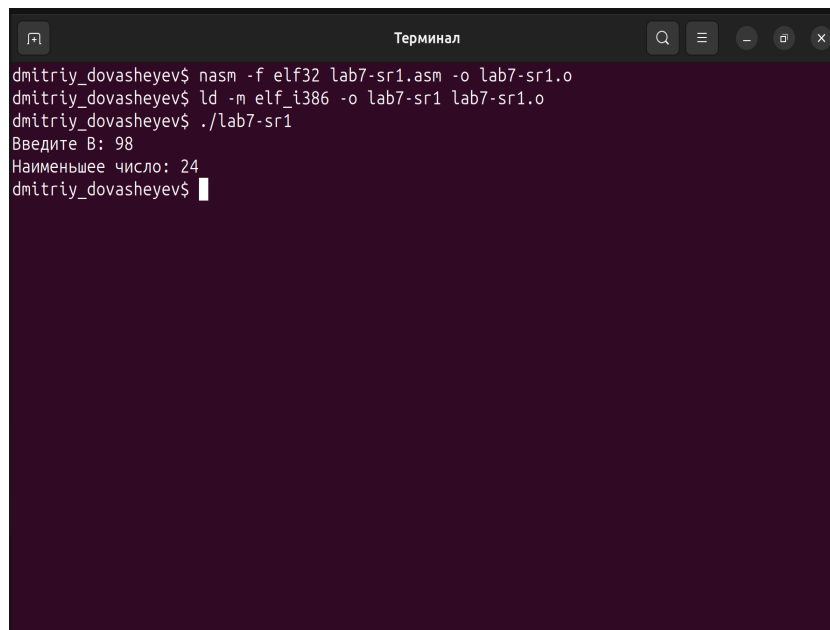
```
call sprint
```

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

```
call iprintLF
```

```
call quit
```

Проверяю корректность написания первой программы (рис. 4.14).

A screenshot of a terminal window titled "Терминал". The terminal shows the following commands and output:

```
dmitriy_dovasheyev$ nasm -f elf32 lab7-sr1.asm -o lab7-sr1.o
dmitriy_dovasheyev$ ld -m elf_i386 -o lab7-sr1 lab7-sr1.o
dmitriy_dovasheyev$ ./lab7-sr1
Введите B: 98
Наименьшее число: 24
dmitriy_dovasheyev$
```

Рис. 4.14: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатуры 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
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        ; 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, res
    call sprint
    mov eax, edi
    call iprintf
    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

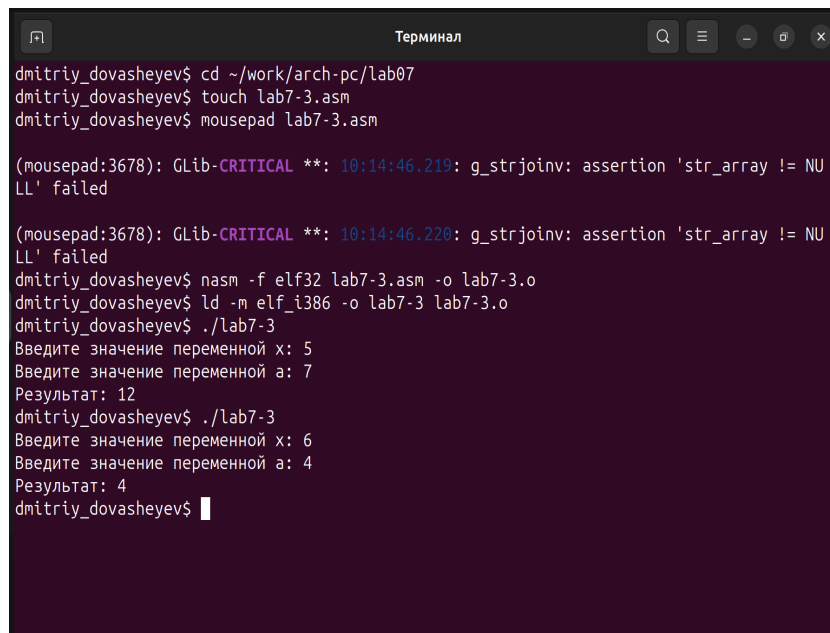
add_values:
mov eax, edi
add eax, esi

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.