

Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера

**Джуманиязов Сарварбек
НКАбд-03-25**

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

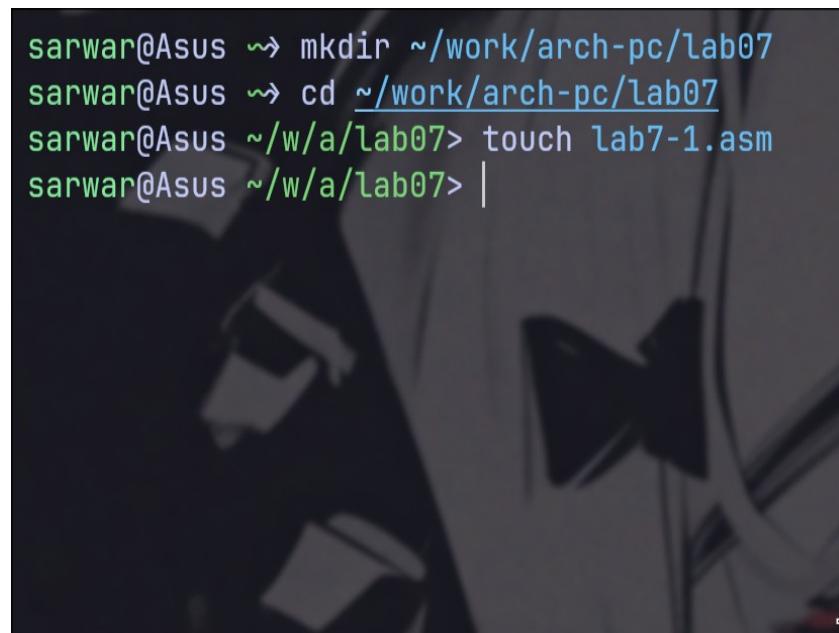
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- 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

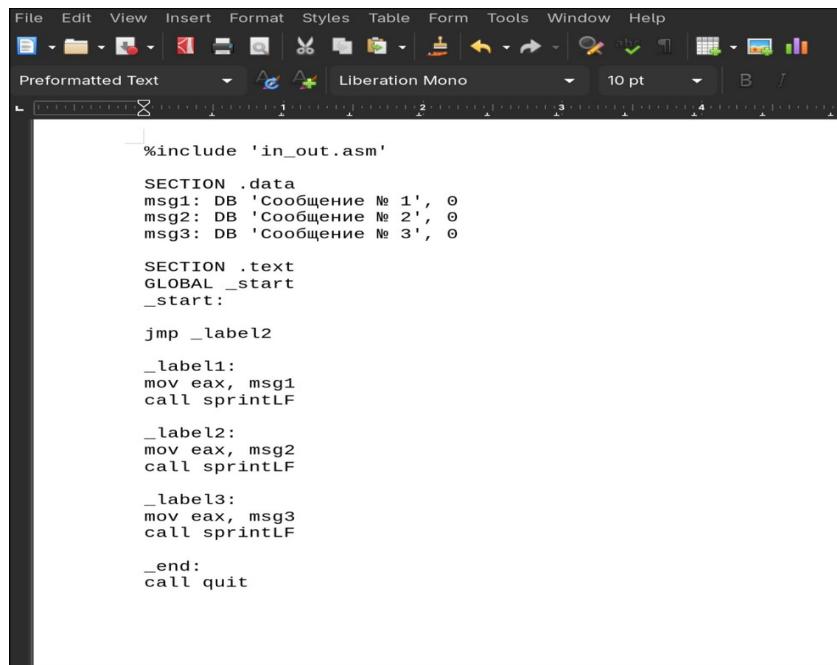
Создаю каталог для программ лабораторной работы №7 (рис. 4.1).



```
sarwar@Asus ~> mkdir ~/work/arch-pc/lab07
sarwar@Asus ~> cd ~/work/arch-pc/lab07
sarwar@Asus ~/w/a/lab07> touch lab7-1.asm
sarwar@Asus ~/w/a/lab07> |
```

Рис. 4.1: Создание каталога и файла для программы

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



```
%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 _label2

_label1:
    mov eax, msg1
    call sprintLF

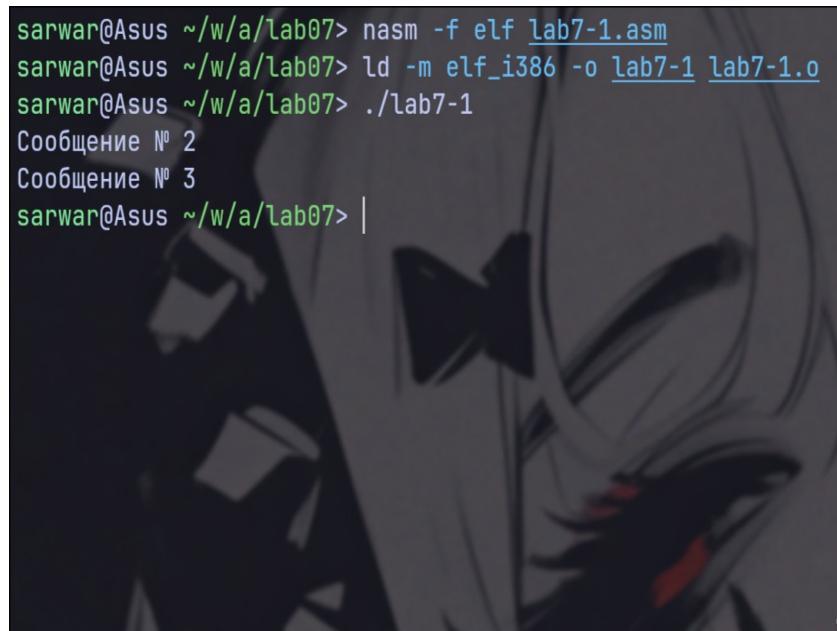
_label2:
    mov eax, msg2
    call sprintLF

_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).

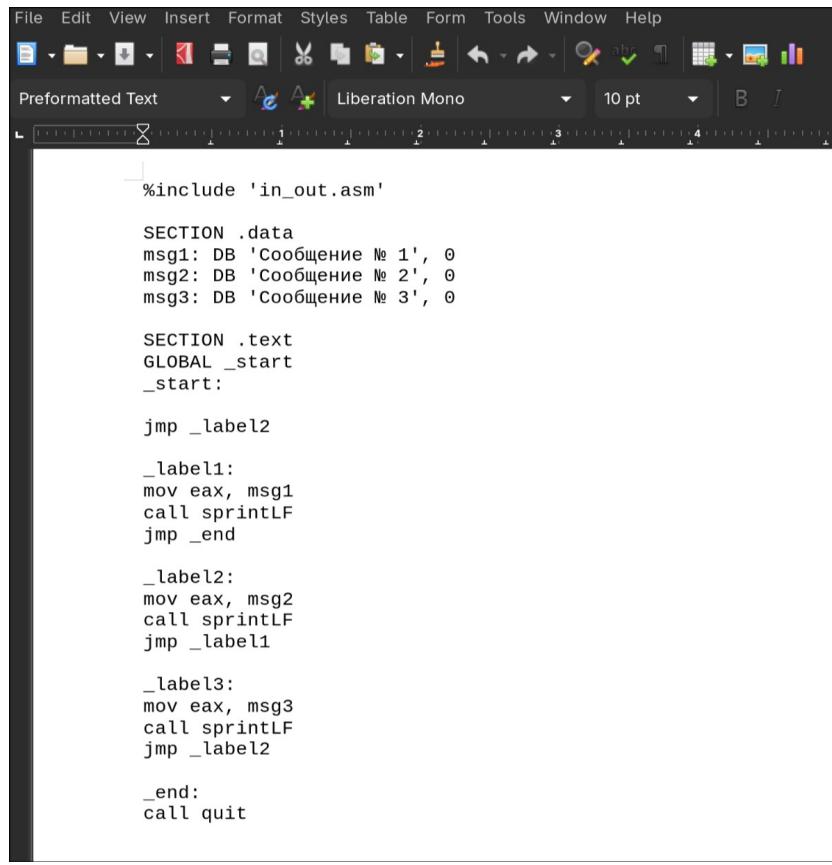


```
sarwar@Asus ~/w/a/lab07> nasm -f elf lab7-1.asm
sarwar@Asus ~/w/a/lab07> ld -m elf_i386 -o lab7-1 lab7-1.o
sarwar@Asus ~/w/a/lab07> ./lab7-1
Сообщение № 2
Сообщение № 3
sarwar@Asus ~/w/a/lab07> |
```

Рис. 4.3: Запуск программы

Изменяю программу таким образом, чтобы поменялся порядок выполнения

функций (рис. 4.4).



The screenshot shows a text editor window with a dark theme. The menu bar includes File, Edit, View, Insert, Format, Styles, Table, Form, Tools, Window, and Help. The toolbar contains various icons for file operations like Open, Save, Print, and Find. The font is set to Liberation Mono at 10 pt. The text area contains the following assembly code:

```
%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 sprintLF
    jmp _end

_label2:
    mov eax, msg2
    call sprintLF
    jmp _label1

_label3:
    mov eax, msg3
    call sprintLF
    jmp _label2

_end:
    call quit
```

Рис. 4.4: Изменение программы

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

```
sarwar@Asus ~/w/a/lab07> nasm -f elf lab7-1.asm
sarwar@Asus ~/w/a/lab07> ld -m elf_i386 -o lab7-1 lab7-1.o
sarwar@Asus ~/w/a/lab07> ./lab7-1
Сообщение № 2
Сообщение № 1
sarwar@Asus ~/w/a/lab07> |
```

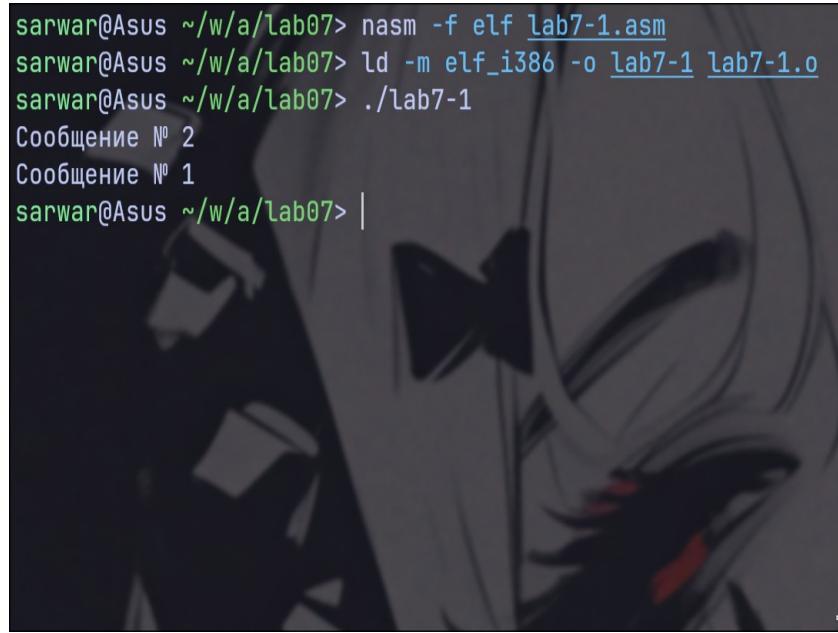
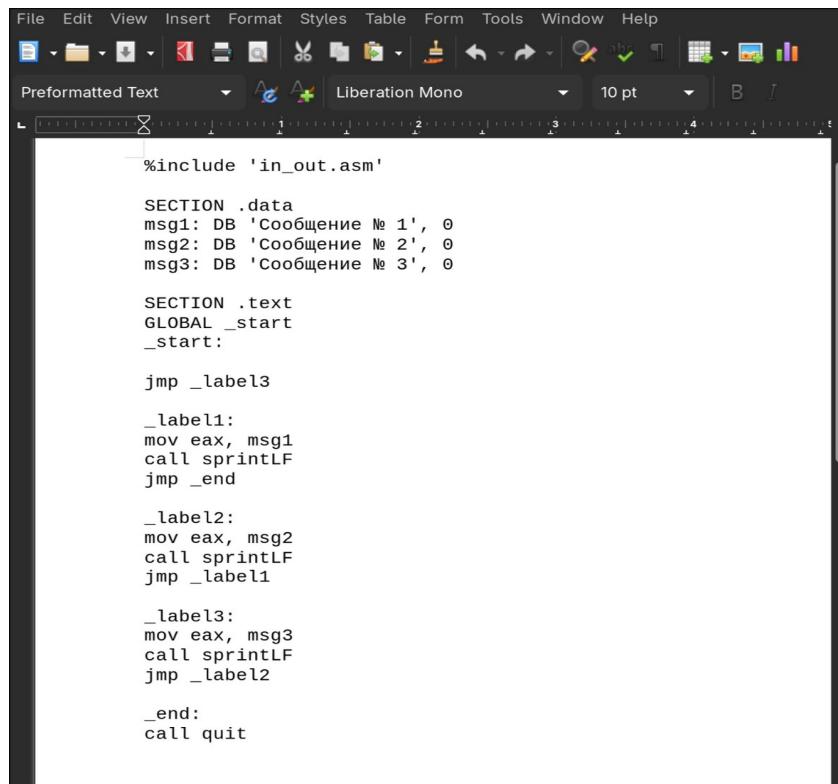


Рис. 4.5: Запуск измененой программы

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



The screenshot shows a Windows Notepad window with the following assembly code:

```
%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).

```
sarwar@Asus ~/w/a/lab07> nasm -f elf lab7-1.asm
sarwar@Asus ~/w/a/lab07> ld -m elf_i386 -o lab7-1 lab7-1.o
sarwar@Asus ~/w/a/lab07> ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
sarwar@Asus ~/w/a/lab07> |
```

Рис. 4.7: Проверка изменений

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

```
%include 'in_out.asm'

SECTION data
msg1 db 'Введите B: ', 0h
msg2 db 'Наименее 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 read
    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
    mov ecx, min
    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 printf
    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).

```
sarwar@Asus ~/w/a/Lab07> touch lab7-2.asm
sarwar@Asus ~/w/a/Lab07> nasm -f elf lab7-2.asm
sarwar@Asus ~/w/a/Lab07> ld -m elf_i386 -o lab7-2 lab7-2.o
sarwar@Asus ~/w/a/Lab07> ./lab7-2
Введите В: 25
Наименьшее число: 15
sarwar@Asus ~/w/a/Lab07> ./lab7-2
Введите В: 60
Наименьшее число: 15
sarwar@Asus ~/w/a/Lab07> ./lab7-2
Введите В: 10
Наименьшее число: 10
sarwar@Asus ~/w/a/Lab07> |
```

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).

```

File Edit Search View Document Help
~/work/arch-pc/lab07/lab7-2.lst - Mousepad
1 1           %include 'in_out.asm'
2 1           <1> ;----- slem -----
3 2           <1> ; функция вычисления длины сообщения
4 3           <1> slem:
5 4 00000000 53   <1> push    ebx
6 5 00000001 89C3  <1> mov     ebx, eax
7 6           <1>
8 7           <1> nextchar:
9 8 00000003 803B00  <1> cmp    byte [eax], 0
10 9 00000005 7403  <1> jz     finished
11 10 00000006 40  <1> inc    eax
12 11 00000009 EBF8  <1> jmp    nextchar
13 12
14 13           <1> finished:
15 14 00000008 2908  <1> sub    eax, ebx
16 15 00000000 58  <1> pop    ebx
17 16 0000000E C3  <1> ret
18 17
19 18
20 19           <1> ;----- sprint -----
21 20           <1> ; функция печати сообщения
22 21           <1> ; входные данные: mov eax, <message>
23 22           <1> sprint:
24 23 0000000F 52   <1> push   edx
25 24 00000010 51  <1> push   ecx
26 25 00000011 53  <1> push   ebx
27 26 00000012 50  <1> push   eax
28 27 00000013 E8EFFFFF  <1> call   slem
29 28
30 29 00000018 89C2  <1> mov    edx, eax
31 30 0000001A 58  <1> pop    eax
32 31
33 32 0000001B 89C1  <1> mov    eax, eax
34 33 0000001D B801000000  <1> mov    ebx, 1
35 34 00000022 B804000000  <1> mov    eax, 4
36 35 00000027 CD80  <1> int    80h
37 36
38 37 00000029 58  <1> pop    ebx
39 38 0000002A 59  <1> pop    ecx
40 39 0000002B 5A  <1> pop    edx
41 40 0000002C C3  <1> ret
42 41
43 42
44 43           <1> ;----- sprintLF -----
45 44           <1> ; функция печати сообщения с переводом строки
46 45           <1> ; входные данные: mov eax, <message>
47 46           <1> sprintLF:
48 47 0000002D E8DFFFFFFF  <1> call   sprint
49 48
50 49 00000032 50  <1> push   eax

```

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).

```

File Edit Search View Document Help
1.1 8 resb 10
1.2
1.3 SECTION .text
1.4 GLOBAL _start
1.5 _start:
1.6
1.7 mov eax, msg1
1.8 call sprint
1.9
1.20 mov ecx, B
1.21 mov edx, 10
1.22 call sread
1.23
1.24 mov eax, B
1.25 call atoi
1.26 mov [B], eax
1.27
1.28 mov ecx, [A]
1.29 mov [max], ecx
1.30
1.31 cmp ecx, [C]
1.32 jg check_B
1.33 mov ecx, [C]
1.34 mov [max], ecx
1.35
1.36 check_B:
1.37 mov eax, |
1.38 call atoi
1.39 mov [max], eax
1.40
1.41 mov ecx, [max]
1.42 cmp ecx, [B]
1.43 jg fin
1.44 mov ecx, [B]
1.45 mov [max], ecx
1.46
1.47 fin:
1.48 mov eax, msg2
1.49 call sprint
1.50 mov eax, [max]
1.51 call iprintLF
1.52 call quit
1.53
1.54
1.55
1.56
1.57

```

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).

```

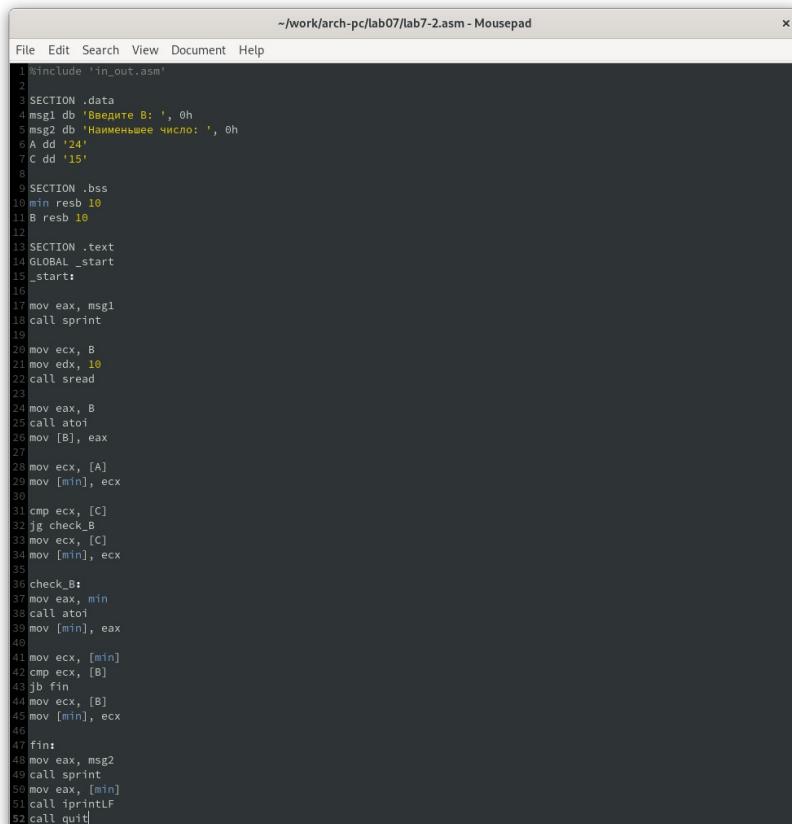
File Edit Search View Document Help
1.1 18 000000ED E81DFFFFFF
1.2
1.3 19
1.4 20 000000F2 B9[0A000000]
1.5 mov ecx, B
1.6 21 000000F7 8A0A000000
1.7 mov edx, 10
1.8 call sread
1.9
1.20 22 000000FC E842FFFF
1.21
1.22 23 000000101 B8[0A000000]
1.23 mov eax, B
1.24 24 000000106 E801FFFF
1.25 call atoi
1.26 25 000000106 E801FFFF
1.27 mov [B], eax
1.28
1.29 26 000000108 A3[0A000000]
1.30 mov [max], eax
1.31
1.32 27
1.33
1.34 28 000000110 B8D136000000
1.35 mov ecx, [A]
1.36 29 000000116 8900[00000000]
1.37 mov [max], ecx
1.38
1.39 30
1.40 31 00000011C 3B0013A0000000
1.41 cmp ecx, [C]
1.42 32 000000122 7F0C
1.43 jg check_B
1.44 33 000000124 880013A0000000
1.45 mov ecx, [C]
1.46 34 00000012A 8900[00000000]
1.47 mov [max], ecx
1.48
1.49 35
1.50
1.51 36
1.52 check_B:
1.53 mov eax, |
1.54 call atoi
1.55
1.56 37 **** Invalid combination of opcode and operands
1.57 38 000000120 E867FFFF
1.58 call atoi
1.59 39 000000135 A3[00000000]
1.60 mov [max], eax
1.61
1.62 40
1.63 41 00000013A B800[00000000]
1.64 mov ecx, [max]
1.65 42 000000140 3B00[0A000000]
1.66 cmp ecx, [B]
1.67 43 000000146 7F0C
1.68 jg fin
1.69 44 000000148 8800[0A000000]
1.70 mov ecx, [B]
1.71 45 00000014E 8900[00000000]
1.72 mov [max], ecx
1.73
1.74 47
1.75
1.76 48 000000154 B8[14000000]
1.77 mov eax, msg2
1.78 49 000000159 E8B1FFFF
1.79 call sprint
1.80 50 00000015E A1[00000000]
1.81 mov eax, [max]
1.82 51 000000163 E81EFFFF
1.83 call iprintLF
1.84 52 000000168 E86EFFFF
1.85 call quit
1.86
1.87 53
1.88
1.89 54
1.90
1.91 55

```

Рис. 4.12: Просмотр ошибки в файле листинга

4.3 Задания для самостоятельной работы

Искренне не понимаю, какой вариант я должен был получить во время 7 лабораторной работы, поэтому буду использовать свой вариант - девятый - из предыдущей лабораторной работы. Возвращаю операнд к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. 4.13).



```
~/work/arch-pc/lab07/lab7-2.asm - Mousepad
File Edit Search View Document Help
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1 db 'Введите B: ', 0h
5 msg2 db 'Наименьшее число: ', 0h
6 A dd '24'
7 C dd '15'
8
9 SECTION .bss
10 min resb 10
11 B resb 10
12
13 SECTION .text
14 GLOBAL _start
15 _start:
16
17 mov eax, msg1
18 call sprint
19
20 mov ecx, B
21 mov edx, 10
22 call sread
23
24 mov eax, B
25 call atoi
26 mov [B], eax
27
28 mov ecx, [A]
29 mov [min], ecx
30
31 cmp ecx, [C]
32 jg check_B
33 mov ecx, [C]
34 mov [min], ecx
35
36 check_B:
37 mov eax, min
38 call atoi
39 mov [min], eax
40
41 mov ecx, [min]
42 cmp ecx, [B]
43 jb fin
44 mov ecx, [B]
45 mov [min], ecx
46
47 fin:
48 mov eax, msg2
49 call sprint
50 mov eax, [min]
51 call iprintLF
52 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).

```
sarwar@Asus ~/w/a/lab07> nasm -f elf lab7-2.asm
sarwar@Asus ~/w/a/lab07> ld -m elf_i386 -o lab7-2 lab7-2.o
sarwar@Asus ~/w/a/lab07> ./lab7-2
Введите В: 98
Наименьшее число: 15
sarwar@Asus ~/w/a/lab07> |
```

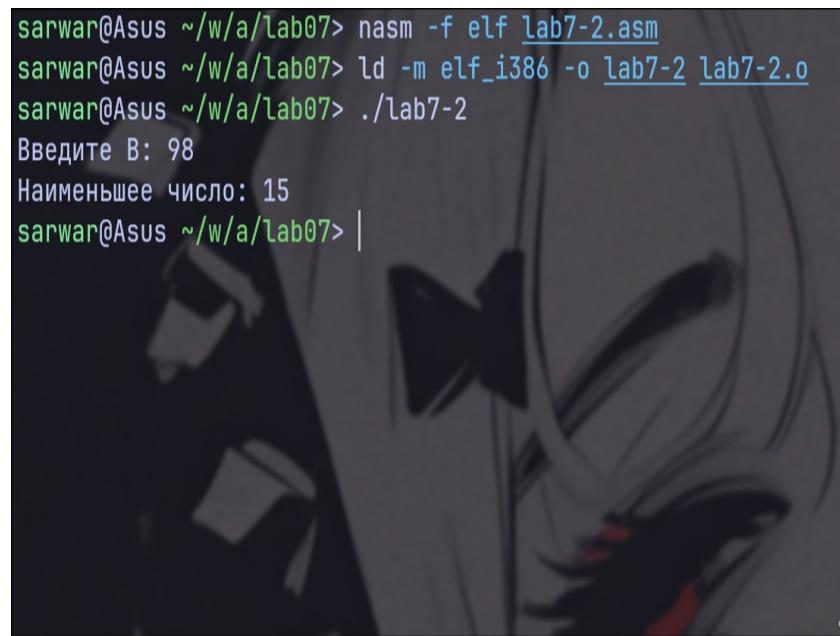


Рис. 4.14: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатурых переменных а и х (рис. 4.15).

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg_x: DB 'Введите значение переменной x: ', 0
4 msg_a: DB 'Введите значение переменной a: ', 0
5 res: DB 'Результат: ', 0
6 SECTION .bss
7 x: RESB 80
8 a: RESB 80
9 SECTION .text
10 GLOBAL _start
11 _start:
12 mov eax, msg_x
13 call sprint
14 mov ecx, x
15 mov edx, $0
16 call sread
17 mov eax, x
18 call atoi
19 mov edi, eax
20
21 mov eax, msg_a
22 call sprint
23 mov ecx, a
24 mov edx, $0
25 call sread
26 mov eax, a
27 call atoi
28 mov esi, eax
29
30 cmp edi, esi
31 jle add_values
32 mov eax, esi
33 jmp print_result
34
35 add_values:
36 mov eax, edi
37 add eax, esi
38
39 print_result:
40 mov edi, eax
41 mov eax, $0
42 call sprint
43 mov eax, edi
44 call iprintLF
45 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:

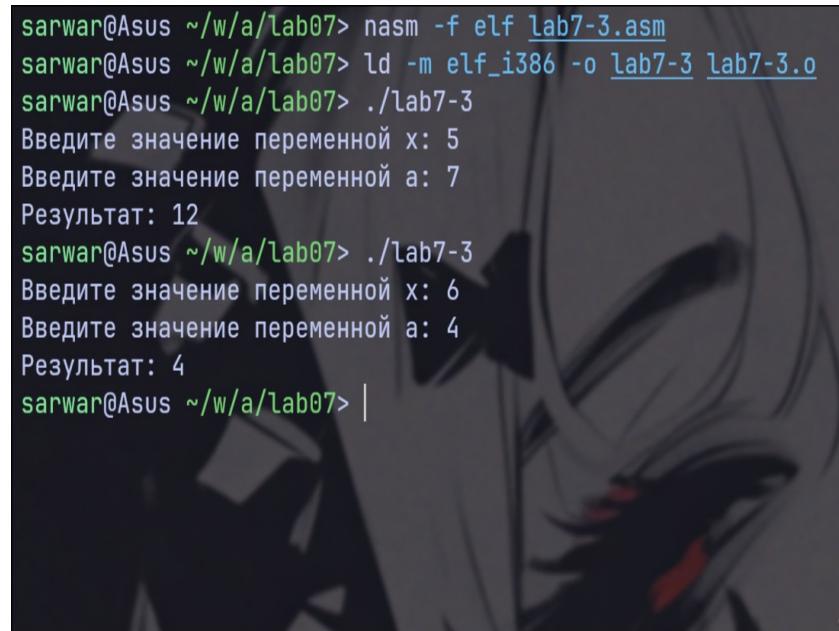
```
mov edi, eax
```

```
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).



```
sarwar@Asus ~/w/a/lab07> nasm -f elf lab7-3.asm  
sarwar@Asus ~/w/a/lab07> ld -m elf_i386 -o lab7-3 lab7-3.o  
sarwar@Asus ~/w/a/lab07> ./lab7-3  
Введите значение переменной x: 5  
Введите значение переменной a: 7  
Результат: 12  
sarwar@Asus ~/w/a/lab07> ./lab7-3  
Введите значение переменной x: 6  
Введите значение переменной a: 4  
Результат: 4  
sarwar@Asus ~/w/a/lab07> |
```

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.