

Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера

Осипов Никита Александрович

Содержание

1	Цель работы	1
2	Задание.....	1
3	Теоретическое введение.....	1
4	Выполнение лабораторной работы	2
4.1	Реализация переходов в NASM.....	2
4.2	Изучение структуры файла листинга.....	6
4.3	Задания для самостоятельной работы.....	7
5	Выводы	12
	Список литературы.....	12

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

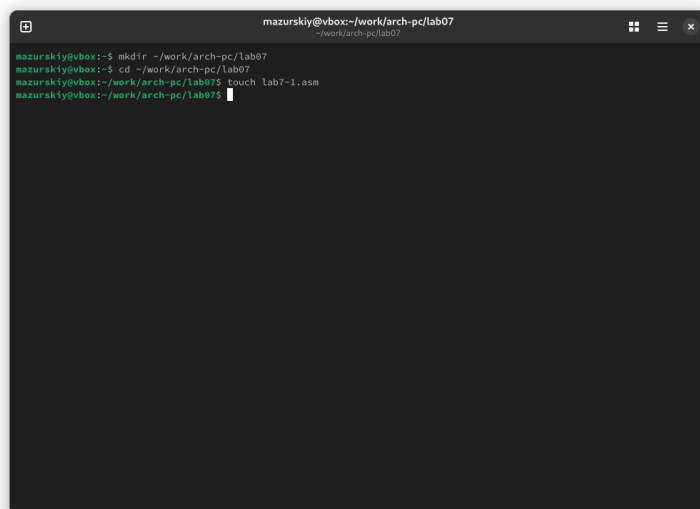
3 Теоретическое введение

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

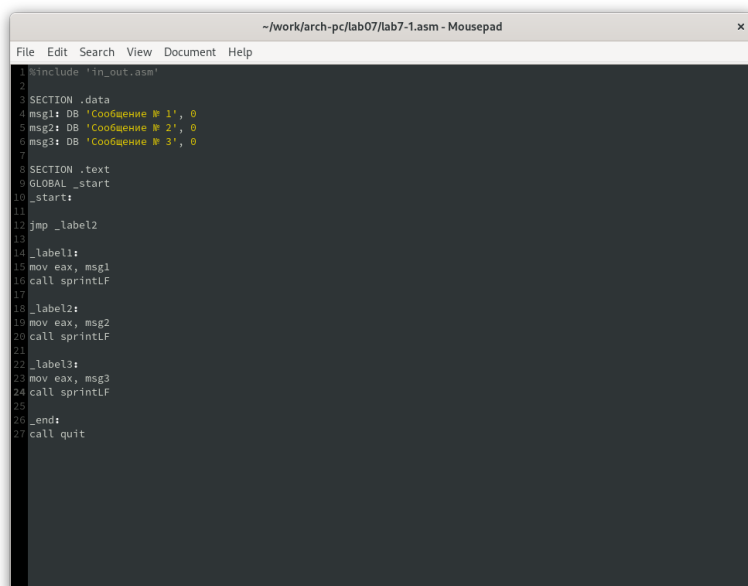
Создаю каталог для программ лабораторной работы №7 (рис. 1).



```
mazurskiy@vbox:~/work/arch-pc/lab07
mazurskiy@vbox:~$ mkdir ~/work/arch-pc/lab07
mazurskiy@vbox:~$ cd ~/work/arch-pc/lab07
mazurskiy@vbox:~/work/arch-pc/lab07$ touch lab7-1.asm
mazurskiy@vbox:~/work/arch-pc/lab07$
```

Рис. 1: Создание каталога и файла для программы

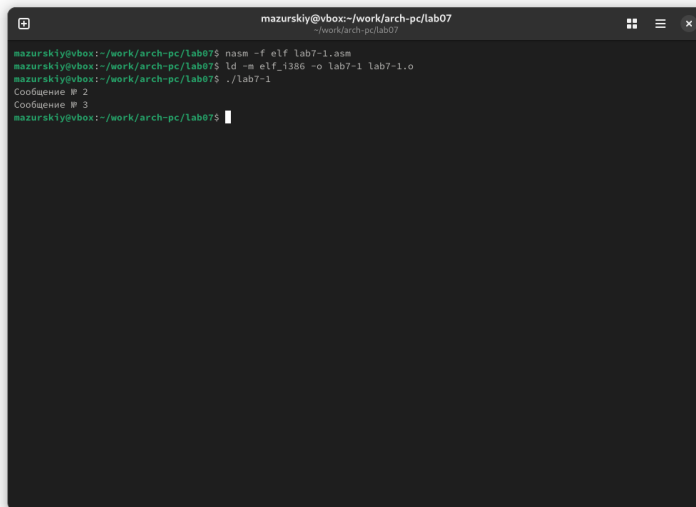
Копирую код из листинга в файл будущей программы. (рис. 2).



```
~/work/arch-pc/lab07/lab7-1.asm - Mousepad
File Edit Search View Document Help
1 include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1', 0
5 msg2: DB 'Сообщение № 2', 0
6 msg3: DB 'Сообщение № 3', 0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 jmp _label2
13
14 _label1:
15 mov eax, msg1
16 call sprintf
17
18 _label2:
19 mov eax, msg2
20 call sprintf
21
22 _label3:
23 mov eax, msg3
24 call sprintf
25
26 _end:
27 call quit
```

Рис. 2: Сохранение программы

При запуске программы я убедился в том, что безусловный переход действительно изменяет порядок выполнения инструкций (рис. 3).

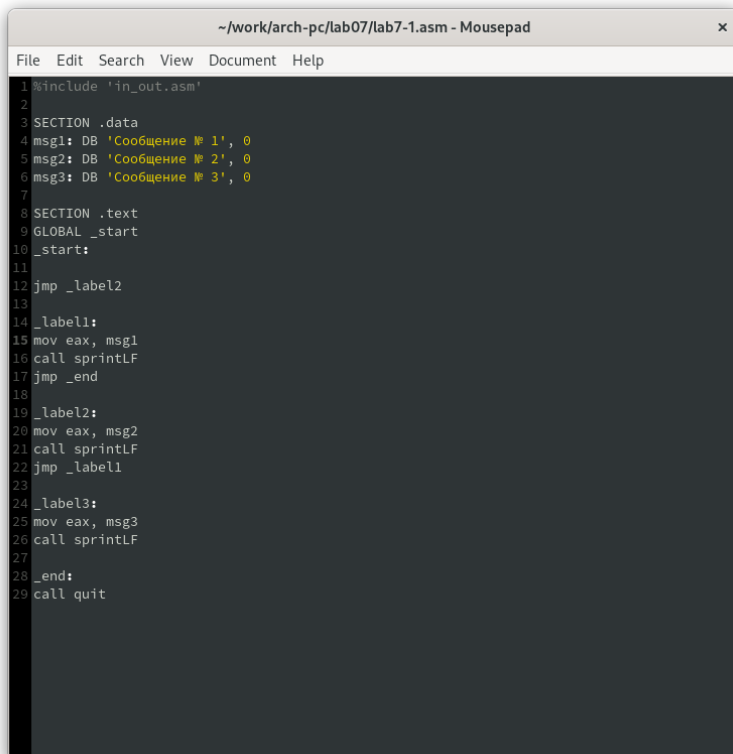


```
mazurskiy@vbox:~/work/arch-pc/lab07
~/work/arch-pc/lab07

mazurskiy@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
mazurskiy@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
mazurskiy@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
mazurskiy@vbox:~/work/arch-pc/lab07$
```

Рис. 3: Запуск программы

Изменяю программу таким образом, чтобы поменялся порядок выполнения функций (рис. 4).

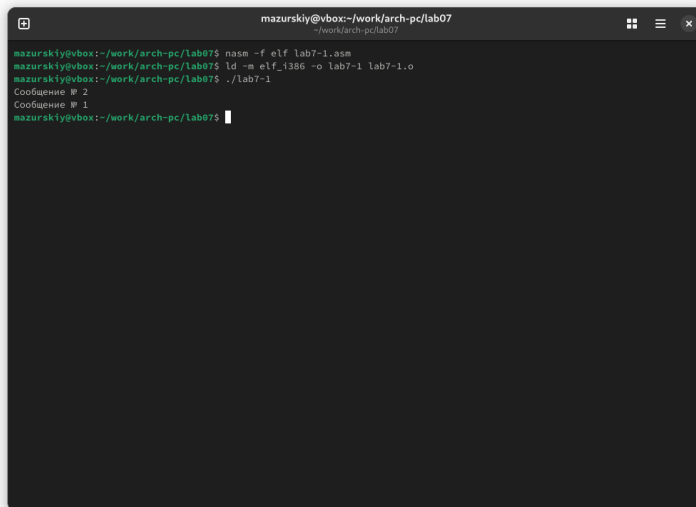


```
~/work/arch-pc/lab07/lab7-1.asm - Mousepad
File Edit Search View Document Help

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1', 0
5 msg2: DB 'Сообщение № 2', 0
6 msg3: DB 'Сообщение № 3', 0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 jmp _label2
13
14 _label1:
15 mov eax, msg1
16 call sprintf
17 jmp _end
18
19 _label2:
20 mov eax, msg2
21 call sprintf
22 jmp _label1
23
24 _label3:
25 mov eax, msg3
26 call sprintf
27
28 _end:
29 call quit
```

Рис. 4: Изменение программы

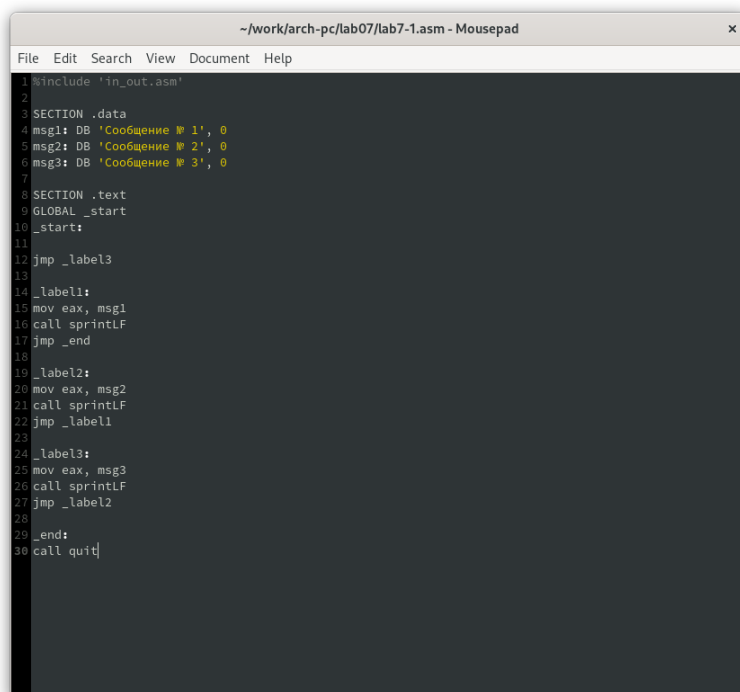
Запускаю программу и проверяю, что примененные изменения верны (рис. 5).



```
mazurskiy@vbox:~/work/arch-pc/lab07
~/work/arch-pc/lab07
mazurskiy@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
mazurskiy@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
mazurskiy@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
Сообщение № 3
mazurskiy@vbox:~/work/arch-pc/lab07$
```

Рис. 5: Запуск измененной программы

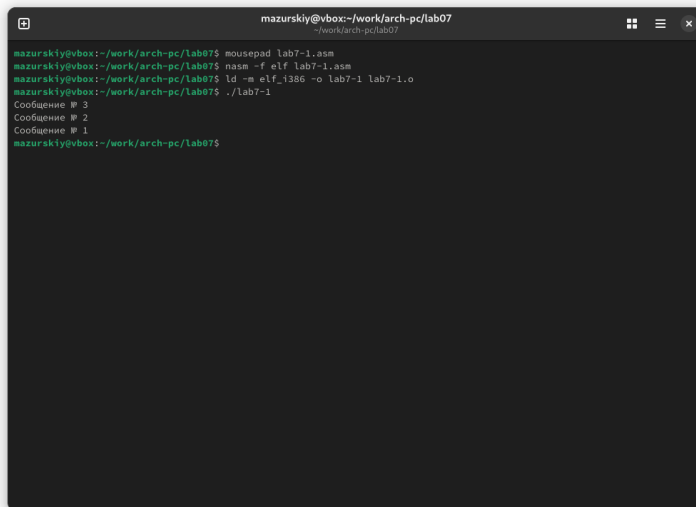
Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. 6).



```
~/work/arch-pc/lab07/lab7-1.asm - Mousepad
File Edit Search View Document Help
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1', 0
5 msg2: DB 'Сообщение № 2', 0
6 msg3: DB 'Сообщение № 3', 0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 jmp _label3
13
14 _label1:
15 mov eax, msg1
16 call sprintf
17 jmp _end
18
19 _label2:
20 mov eax, msg2
21 call sprintf
22 jmp _label1
23
24 _label3:
25 mov eax, msg3
26 call sprintf
27 jmp _label2
28
29 _end:
30 call quit
```

Рис. 6: Изменение программы

Работа выполнена корректно, программа в нужном мне порядке выводит сообщения (рис. 7).

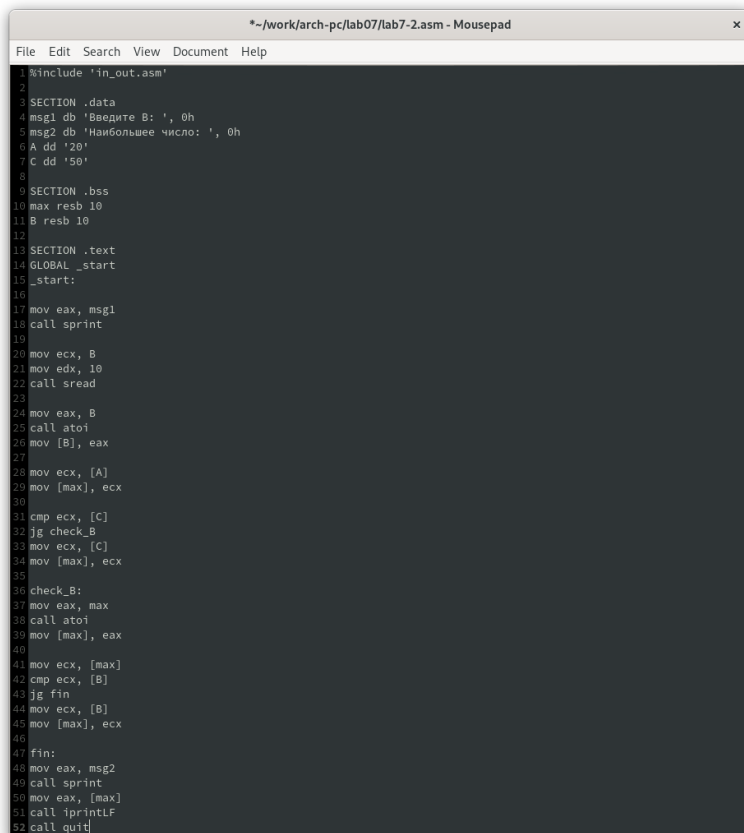


```
mazurskiy@vbox:~/work/arch-pc/lab07
~/work/arch-pc/lab07

mazurskiy@vbox:~/work/arch-pc/lab07$ mousepad lab7-1.asm
mazurskiy@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
mazurskiy@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
mazurskiy@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение # 3
Сообщение # 2
Сообщение # 1
mazurskiy@vbox:~/work/arch-pc/lab07$
```

Рис. 7: Проверка изменений

Создаю новый рабочий файл и вставляю в него код из следующего листинга (рис. 8).



```
*~/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 'Наибольшее число: ', 0h
6 A dd '20'
7 C dd '50'
8
9 SECTION .bss
10 max 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 [max], ecx
30
31 cmp ecx, [C]
32 jg check_B
33 mov ecx, [C]
34 mov [max], ecx
35
36 check_B:
37 mov eax, max
38 call atoi
39 mov [max], eax
40
41 mov ecx, [max]
42 cmp ecx, [B]
43 jg fin
44 mov ecx, [B]
45 mov [max], ecx
46
47 fin:
48 mov eax, msg2
49 call sprint
50 mov eax, [max]
51 call 'printf'
52 call quit
```

Рис. 8: Сохранение новой программы

Программа выводит значение переменной с максимальным значением, проверяю работу программы с разными входными данными (рис. 9).

```
mazurskiy@vbox:~/work/arch-pc/lab07
~/work/arch-pc/lab07

mazurskiy@vbox:~/work/arch-pc/lab07$ touch lab7-2.asm
mazurskiy@vbox:~/work/arch-pc/lab07$ mousepad lab7-2.asm
mazurskiy@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
mazurskiy@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
mazurskiy@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 25
Наибольшее число: 50
mazurskiy@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
mazurskiy@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
mazurskiy@vbox:~/work/arch-pc/lab07$
```

Рис. 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. 10).

```
File Edit Search View Document Help
~/work/arch-pc/lab07/lab7-2.lst - Mousepad

1      ;include 'in_out.asm'
2      <|> ;----- slen -----
3      <|> ; функция вычисления длины сообщения
4      00000000 53      <|> push    ebx, eax
5      00000001 89C3    <|> mov     ebx, eax
6      7
7      <|> nextchar:
8      00000003 8B1800    <|> cmp     byte [eax], 0
9      00000005 7403      <|> jz      finished
10     00000008 40          <|> inc     eax
11     00000009 EBF8      <|> jmp     nextchar
12
13     <|> finished:
14     0000000B 29D8      <|> sub     eax, ebx
15     0000000D 58          <|> pop     ebx
16     0000000E C3          <|> ret
17
18     <|>
19     <|> ;----- sprintf -----
20     <|> ; функция печати сообщения
21     <|> ; входные данные: mov eax, message
22     <|> sprintf:
23     0000000F 52          <|> push    edx
24     00000010 51          <|> push    ecx
25     00000011 53          <|> push    ebx
26     00000012 58          <|> push    eax
27     00000013 E8C0FFFF    <|> call    slen
28     00000014
29     00000015 89C2      <|> mov     edx, eax
30     00000016 58          <|> pop     eax
31     00000017
32     00000018 89C1      <|> mov     ecx, eax
33     00000019 8B01000000 <|> mov     ebx, 1
34     0000001A 8B04000000 <|> mov     eax, 4
35     0000001B CDB0      <|> int     80h
36     0000001C
37     0000001D 58          <|> pop     ebx
38     0000001E 59          <|> pop     ecx
39     0000001F 5A          <|> pop     edx
40     00000020 C3          <|> ret
41
42     <|>
43     <|> ;----- sprintf -----
44     <|> ; функция печати сообщения с переводом строки
45     <|> ; входные данные: mov eax, message
46     <|> sprintf:
47     0000002D E8D0FFFF    <|> call    sprintf
48     00000030
49     00000031 50          <|> push    eax
```

Рис. 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. 11).

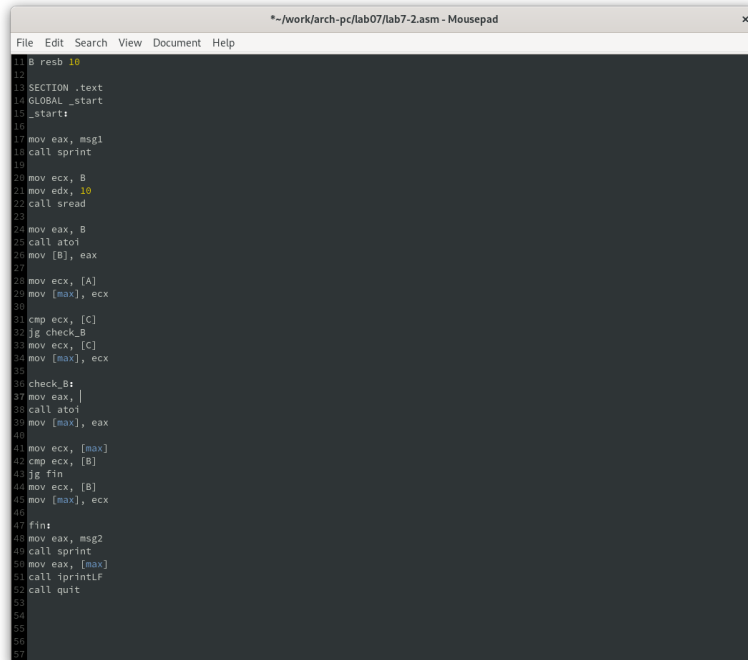


Рис. 11: Удаление операнда из программы

В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются. (рис. 12).

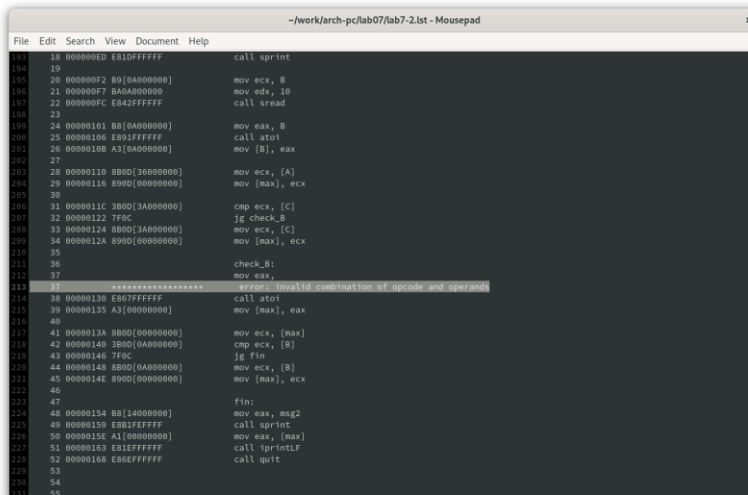
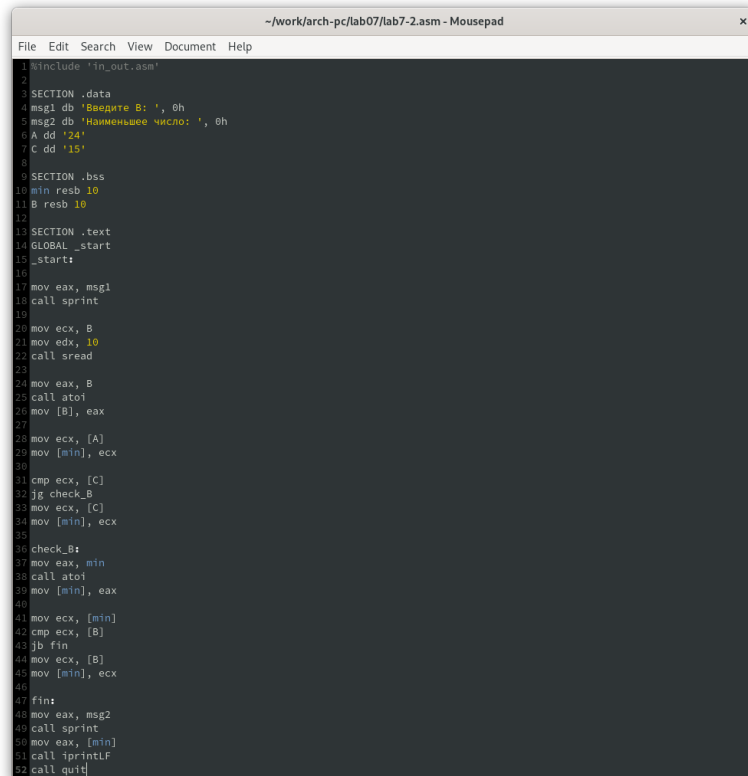


Рис. 12: Просмотр ошибки в файле листинга

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. 13).



```
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 iprintf
52 call quit
```

Рис. 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. 14).

```
mazurskiy@vbox:~/work/arch-pc/lab07
~/work/arch-pc/lab07

mazurskiy@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
mazurskiy@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
mazurskiy@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 98
Наименьшее число: 15
mazurskiy@vbox:~/work/arch-pc/lab07$
```

Рис. 14: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатуры переменных а и х (рис. 15).

```
~/work/arch-pc/lab07/lab7-3.asm - Mousepad
File Edit Search View Document Help

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, 80
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, 80
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, res
42 call sprint
43 mov eax, edi
44 call iprintf
45 call quit
```

Рис. 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

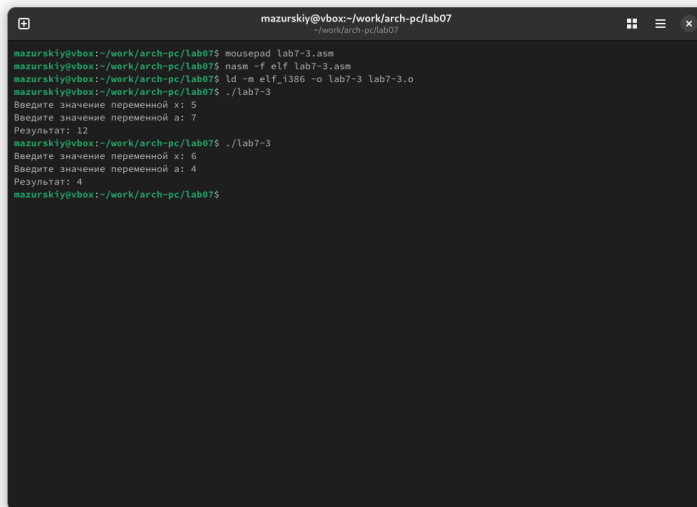
cmp edi, esi
jle add_values
mov eax, esi
jmp print_result

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. 16).

A screenshot of a terminal window with a dark background. The window title is "mazurskiy@vbox:~/work/arch-pc/lab07". The terminal shows the following commands and output:

```
mazurskiy@vbox:~/work/arch-pc/lab07$ mousepad lab7-3.asm
mazurskiy@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
mazurskiy@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
mazurskiy@vbox:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 5
Введите значение переменной a: 7
Результат: 12
mazurskiy@vbox:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 6
Введите значение переменной a: 4
Результат: 4
mazurskiy@vbox:~/work/arch-pc/lab07$
```

Рис. 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.