

# Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера

Хан Георгий Игоревич

# Оглавление

|  |    |
|--|----|
| 1 Цель работы .....                          | 2  |
| 2 Задание .....                              | 2  |
| 3 Теоретическое введение .....               | 2  |
| 4 Выполнение лабораторной работы .....       | 3  |
| 4.1 Реализация переходов в NASM .....        | 3  |
| 4.2 Изучение структуры файла листинга.....   | 8  |
| 4.3 Задания для самостоятельной работы ..... | 9  |
| 5 Выводы.....                                | 13 |
| Список литературы.....                       | 13 |

## 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

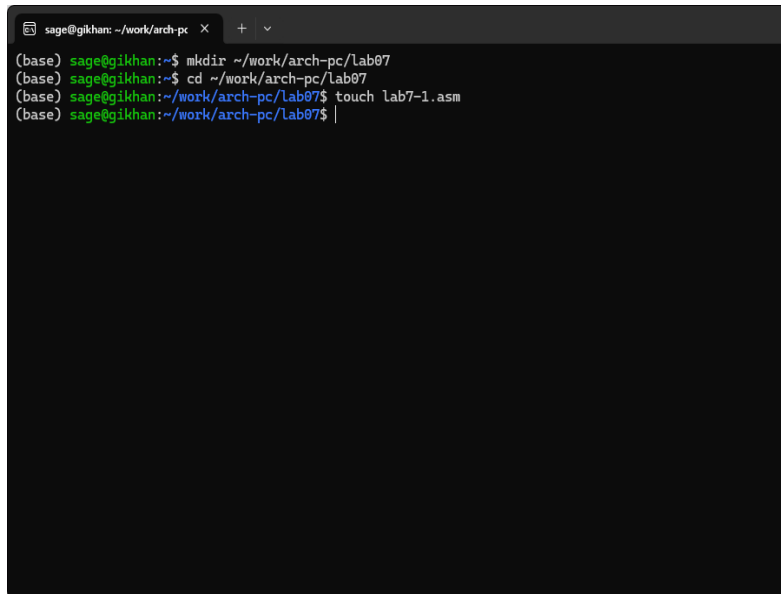
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- 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

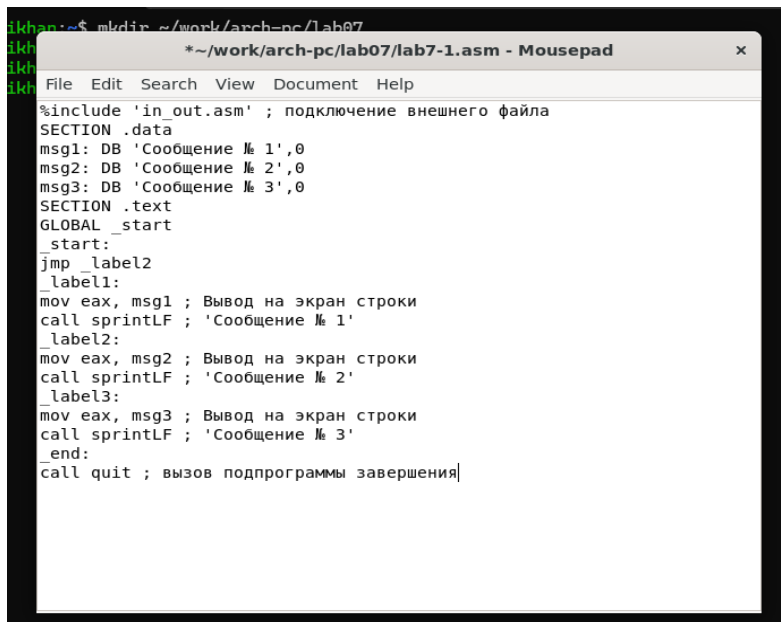
Создаю каталог для программ лабораторной работы №7 (рис. 1).



```
sage@gikhan: ~/work/arch-pc x + v
(base) sage@gikhan:~$ mkdir ~/work/arch-pc/lab07
(base) sage@gikhan:~$ cd ~/work/arch-pc/lab07
(base) sage@gikhan:~/work/arch-pc/lab07$ touch lab7-1.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ |
```

Рис. 1: Создание каталога и файла для программы

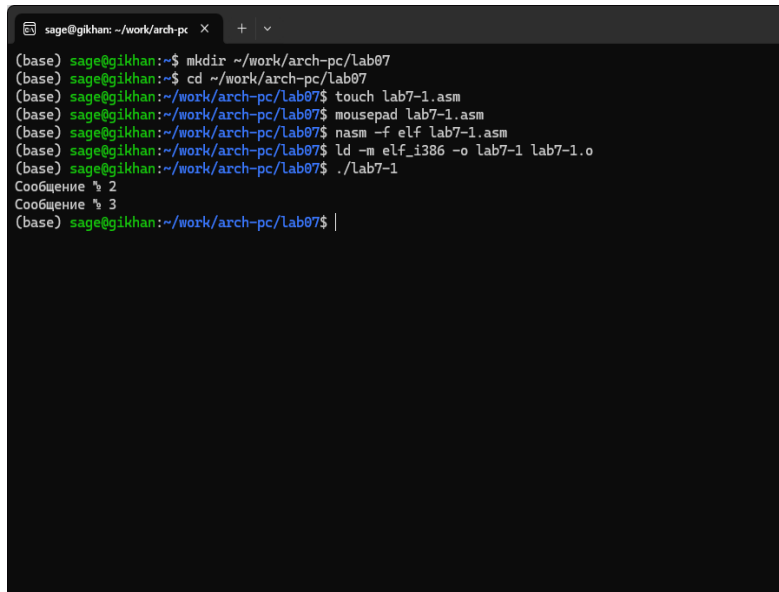
Копирую код из листинга в файл будущей программы. (рис. 2).



```
ikhan:~$ mkdir ~/work/arch-pc/lab07
ikhan:~$ cd ~/work/arch-pc/lab07
ikhan:~/work/arch-pc/lab07$ cat lab7-1.asm
%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 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения|
```

Рис. 2: Сохранение программы

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

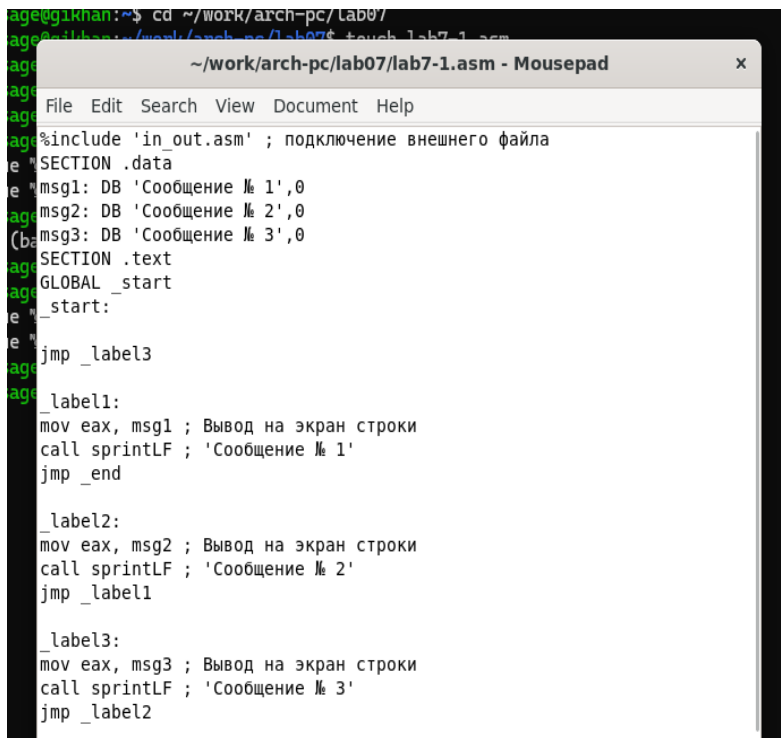
A terminal window with a dark background and light green text. The window title is 'sage@gikhan: ~/work/arch-pc'. The terminal shows a series of commands and their outputs. The commands are: 'mkdir ~/work/arch-pc/lab07', 'cd ~/work/arch-pc/lab07', 'touch lab7-1.asm', 'mousepad lab7-1.asm', 'nasm -f elf lab7-1.asm', 'ld -m elf\_i386 -o lab7-1 lab7-1.o', and './lab7-1'. The outputs are: '(base) sage@gikhan:~\$ mkdir ~/work/arch-pc/lab07', '(base) sage@gikhan:~\$ cd ~/work/arch-pc/lab07', '(base) sage@gikhan:~/work/arch-pc/lab07\$ touch lab7-1.asm', '(base) sage@gikhan:~/work/arch-pc/lab07\$ mousepad lab7-1.asm', '(base) sage@gikhan:~/work/arch-pc/lab07\$ nasm -f elf lab7-1.asm', '(base) sage@gikhan:~/work/arch-pc/lab07\$ ld -m elf\_i386 -o lab7-1 lab7-1.o', and '(base) sage@gikhan:~/work/arch-pc/lab07\$ ./lab7-1'. The last command is followed by two lines of output: 'Сообщение % 2' and 'Сообщение % 3'.

```
(base) sage@gikhan:~$ mkdir ~/work/arch-pc/lab07
(base) sage@gikhan:~$ cd ~/work/arch-pc/lab07
(base) sage@gikhan:~/work/arch-pc/lab07$ touch lab7-1.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-1.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
(base) sage@gikhan:~/work/arch-pc/lab07$ ./lab7-1
Сообщение % 2
Сообщение % 3
(base) sage@gikhan:~/work/arch-pc/lab07$ |
```

*Рис. 3: Запуск программы*

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





```
sage@gikhan:~$ cd ~/work/arch-pc/Lab07/
sage@gikhan:~/work/arch-pc/Lab07$ touch lab7-1.asm
sage@gikhan:~/work/arch-pc/Lab07$ mousepad lab7-1.asm
~/work/arch-pc/lab07/lab7-1.asm - Mousepad
File Edit Search View Document Help
%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 _label3

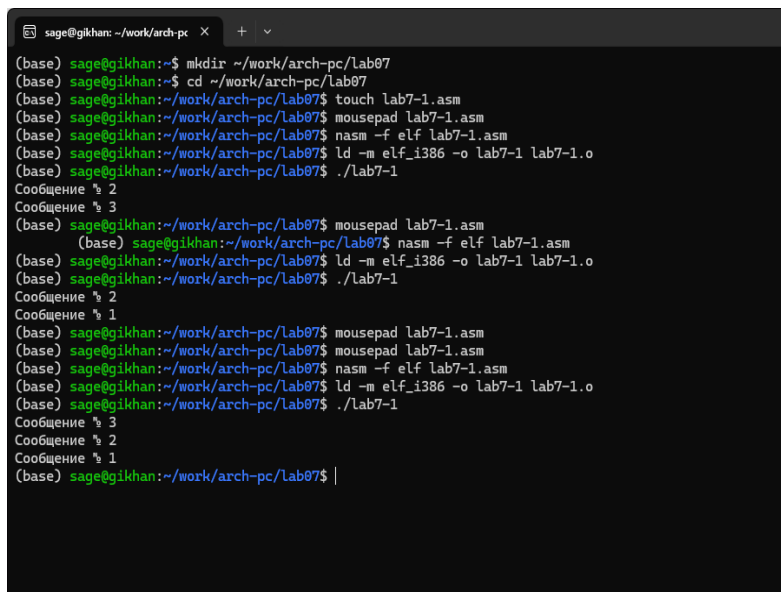
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2
```

Рис. 6: Изменение программы

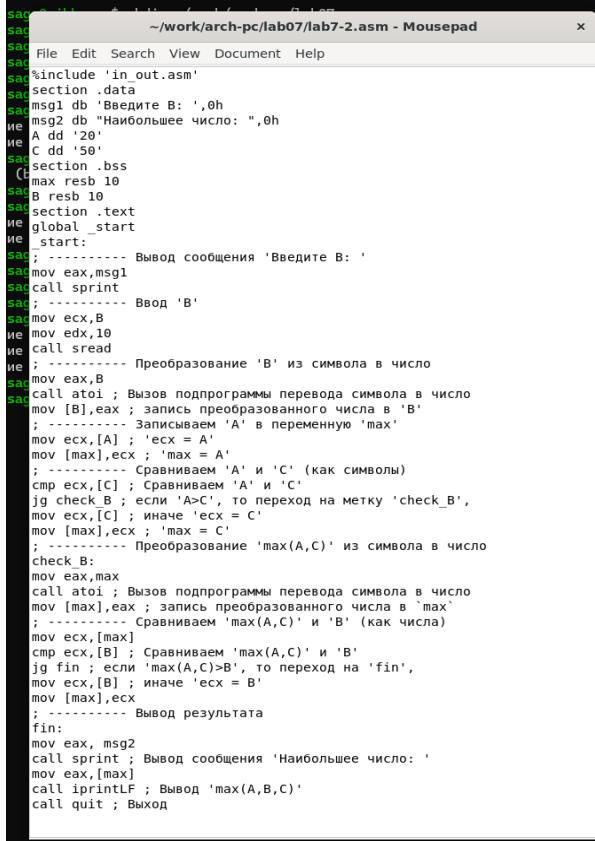
Работа выполнена корректно, программа в нужном мне порядке выводит сообщения (рис. 7).



```
sage@gikhan:~/work/arch-pc$ mkdir ~/work/arch-pc/Lab07
sage@gikhan:~/work/arch-pc/Lab07$ cd ~/work/arch-pc/Lab07
sage@gikhan:~/work/arch-pc/Lab07$ touch lab7-1.asm
sage@gikhan:~/work/arch-pc/Lab07$ mousepad lab7-1.asm
sage@gikhan:~/work/arch-pc/Lab07$ nasm -f elf lab7-1.asm
sage@gikhan:~/work/arch-pc/Lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
sage@gikhan:~/work/arch-pc/Lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
sage@gikhan:~/work/arch-pc/Lab07$ mousepad lab7-1.asm
sage@gikhan:~/work/arch-pc/Lab07$ nasm -f elf lab7-1.asm
sage@gikhan:~/work/arch-pc/Lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
sage@gikhan:~/work/arch-pc/Lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
sage@gikhan:~/work/arch-pc/Lab07$ mousepad lab7-1.asm
sage@gikhan:~/work/arch-pc/Lab07$ mousepad lab7-1.asm
sage@gikhan:~/work/arch-pc/Lab07$ nasm -f elf lab7-1.asm
sage@gikhan:~/work/arch-pc/Lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
sage@gikhan:~/work/arch-pc/Lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
sage@gikhan:~/work/arch-pc/Lab07$ |
```

Рис. 7: Проверка изменений

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



*Рис. 8: Сохранение новой программы*

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

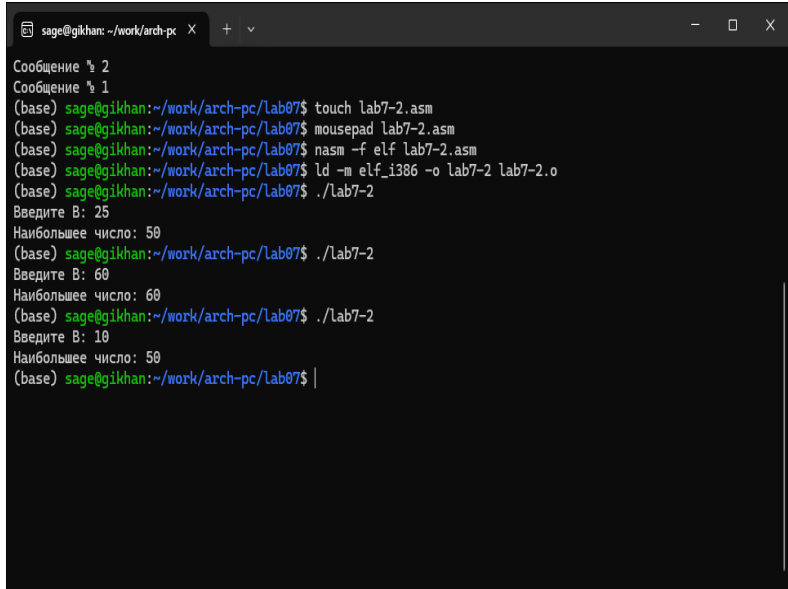


Рис. 9: Проверка программы из листинга

## 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).

```
1      %include 'in_out.asm'
2      ;----- slen -----
3      ; Функция вычисления длины строки
4      slen:
5          push    ebx
6          mov     ebx, eax
7          nextchar:
8          cmp     byte [eax], 0
9          jz      finished
10         inc     eax
11         jmp     nextchar
12     finished:
13         sub     eax, ebx
14         pop     ebx
15         ret
16
17
18
19     ;----- sprintf -----
20     ; Функция печати сообщения
21     ; входные данные: mov eax, <message>
22     sprintf:
23         push    edx
24         push    ecx
25         push    ebx
26         push    eax
27         call   slen
28
29         mov     edx, eax
30         pop     eax
31
32         mov     ecx, eax
33         mov     ebx, 1
34         mov     eax, 4
35         int     80h
36
37         pop     ebx
38         pop     ecx
39         pop     edx
40         ret
41
42
43     ;----- sprintfLF -----
44     ; Функция печати сообщения с переводом строки
45     ; входные данные: mov eax, <message>
46     sprintfLF:
47         call   sprintf
48
49         push    eax
50         mov     eax, 0Ah
51         push    eax
52         call   sprintfLF
53         pop     eax
54         mov     eax, 0
55         ret
56
```

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).



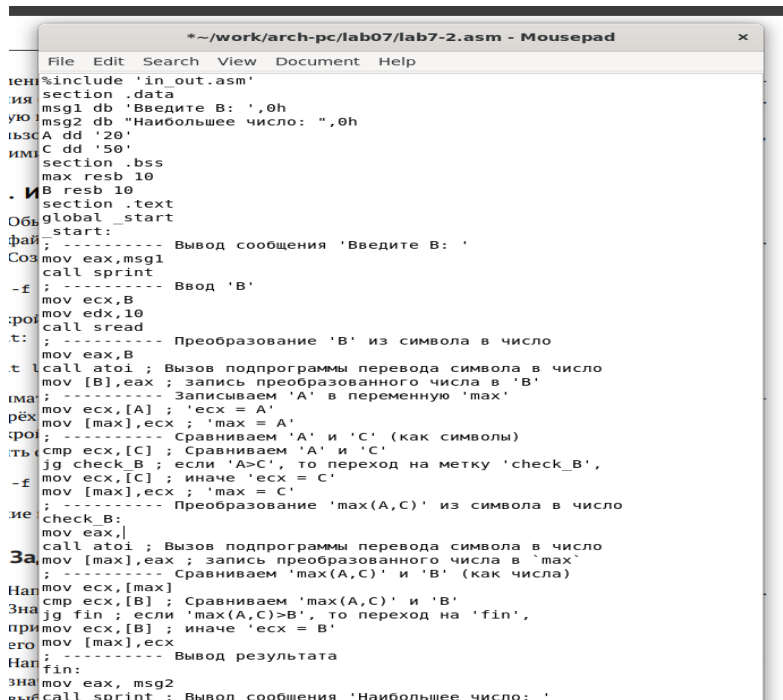


Рис. 11: Удаление операнда из программы

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

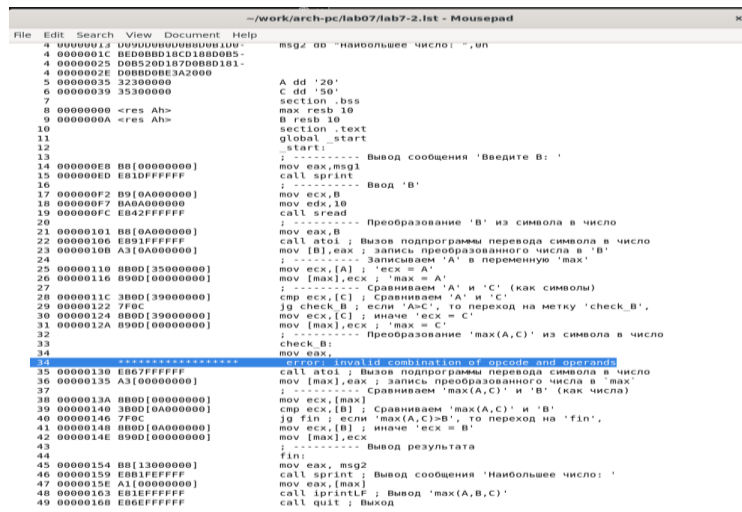
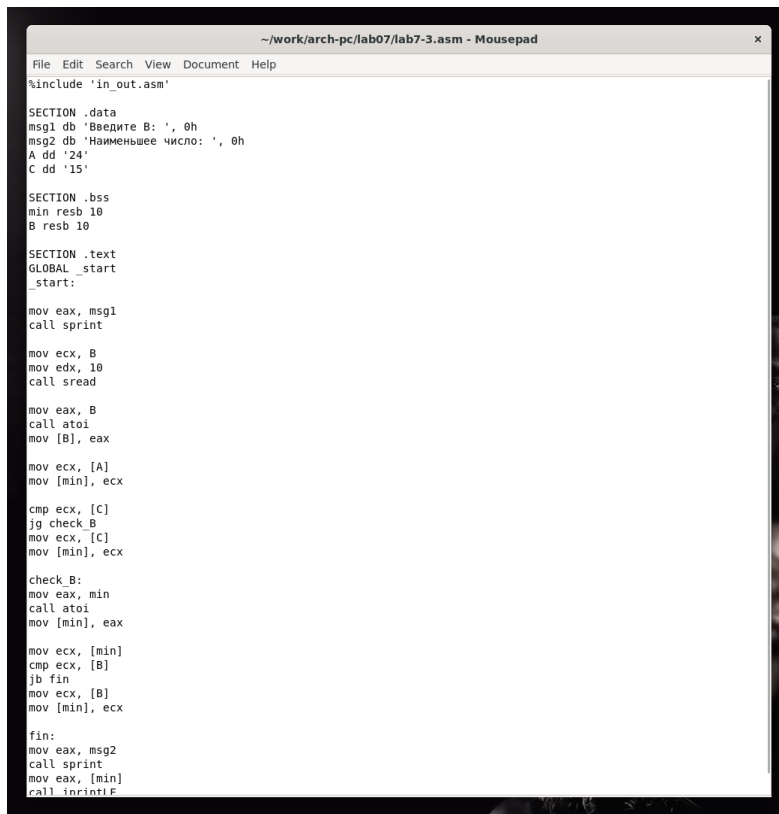


Рис. 12: Просмотр ошибки в файле листинга

## 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).

A screenshot of a text editor window titled '~\work\arch-pc\lab07\lab7-3.asm - Mousepad'. The window contains assembly code for a program that finds the minimum of three numbers. The code includes sections for data, bss, and text. It defines messages, reserves space for variables, and implements logic to read three numbers and compare them to find the minimum. The code is as follows:

```
%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 iprintf
```

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).

```
sage@gikhan: ~/work/arch-pc
(base) sage@gikhan:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
(base) sage@gikhan:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 25
Наибольшее число: 50
(base) sage@gikhan:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
(base) sage@gikhan:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
(base) sage@gikhan:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-2.lst
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-2.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:34: error: invalid combination of opcode and operands
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-2.lst
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-2.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-2.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ touch lab7-3.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-3.asm
x(base) sage@gikhan:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
ld: cannot find lab7-2.o: No such file or directory
(base) sage@gikhan:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
(base) sage@gikhan:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 98
Наименьшее число: 15
(base) sage@gikhan:~/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/lab7-4.asm - Mousepad
File Edit Search View Document Help
%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 print
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax

mov eax, msg_a
call print
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 print
mov eax, edi
call iprintf
call quit
```

Рис. 15: Вторая программа самостоятельной работы

Код второй программы:

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений а и х (рис. 16).

```
sage@gikhan: ~/work/arch-pc x + v
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-2.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:34: error: invalid combination of opcode and operands
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-2.lst
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-2.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-2.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ touch lab7-3.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-3.asm
x(base) sage@gikhan:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
ld: cannot find lab7-2.o: No such file or directory
(base) sage@gikhan:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
(base) sage@gikhan:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 98
Наименьшее число: 15
(base) sage@gikhan:~/work/arch-pc/lab07$ touch lab7-4.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-4.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ mousepad lab7-4.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
(base) sage@gikhan:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
(base) sage@gikhan:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 5
Введите значение переменной a: 7
Результат: 12
(base) sage@gikhan:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 6
Введите значение переменной a: 4
Результат: 4
(base) sage@gikhan:~/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.