

# **Шаблон отчёта по лабораторной работе**

**7**

Баптишта Матеуж

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы :</b>	<b>6</b>
2.1	Изучение структуры файлы листинга :	11
2.2	Выводы по результатам выполнения заданий :	12
<b>3</b>	<b>Задание для самостоятельной работы :</b>	<b>13</b>
3.1	Написание программы нахождения наименьшей из 3 целочислен- ных переменных : .....	13
3.2	Написание программы .....	15
<b>4</b>	<b>Выводы по результатам выполнения заданий :</b>	<b>18</b>
<b>5</b>	<b>Выводы, согласованные с целью работы :</b>	<b>19</b>
	<b>Список литературы</b>	<b>20</b>

## Список иллюстраций

2.1 рисунок.....	Erro! Indicador não definido.
2.2 рисунок.....	6
2.3 рисунок.....	6
2.4 рисунок.....	7
2.5 рисунок.....	7
2.6 рисунок.....	8
2.7 рисунок.....	9
2.8 рисунок.....	10
2.9 рисунок.....	10
2.10 рисунок.....	11
2.11 рисунок.....	11
3.1 рисунок.....	13
3.2 рисунок.....	14
3.3 рисунок.....	15
3.4 рисунок.....	16

## Список таблиц

# 1 Цель работы

- В восьмой лабораторной работе мы узнаем о команде условных и безусловных переходов, делая это, мы освоим использование переходов, а также познакомимся со структурой файла листинга.

## 2 Выполнение лабораторной работы :

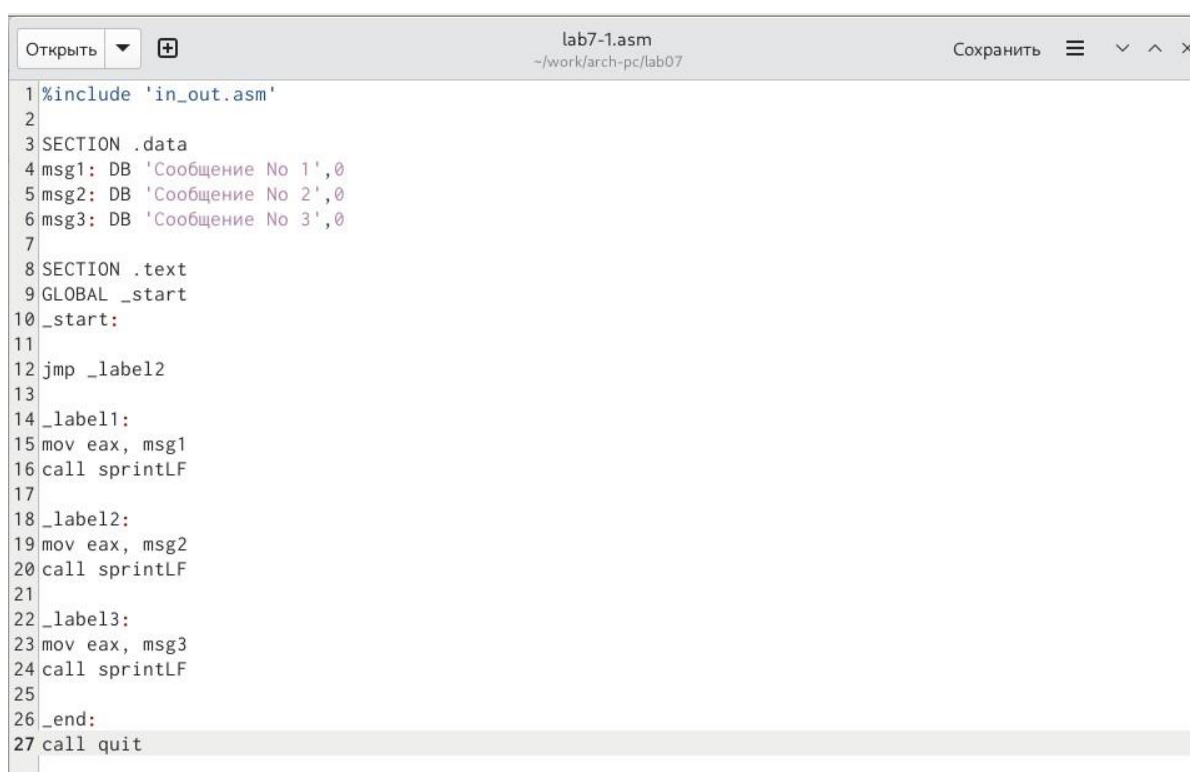
##Реализация переходов в NASM :

- Здесь мы начали с создания, а затем переместились в восьмой каталог лаборатории “~/work/arch-pc/lab07”, после чего мы создали файл “lab7-1.asm”.(рис. [2.1])

```
[mabaptishta@fedora arch-pc]$ mkdir ~/work/arch-pc/lab07
[mabaptishta@fedora arch-pc]$ cd ~/work/arch-pc/lab07
[mabaptishta@fedora lab07]$ touch lab7-1 .asm
[mabaptishta@fedora lab07]$
```

Рис. 2.1: рисунок

- После этого мы заполнили файл .asm кодом программы, отображающей значение регистра eax.(рис. [2.2])



```
lab7-1.asm
~/work/arch-pc/lab07

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение No 1',0
5 msg2: DB 'Сообщение No 2',0
6 msg3: DB 'Сообщение No 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
```

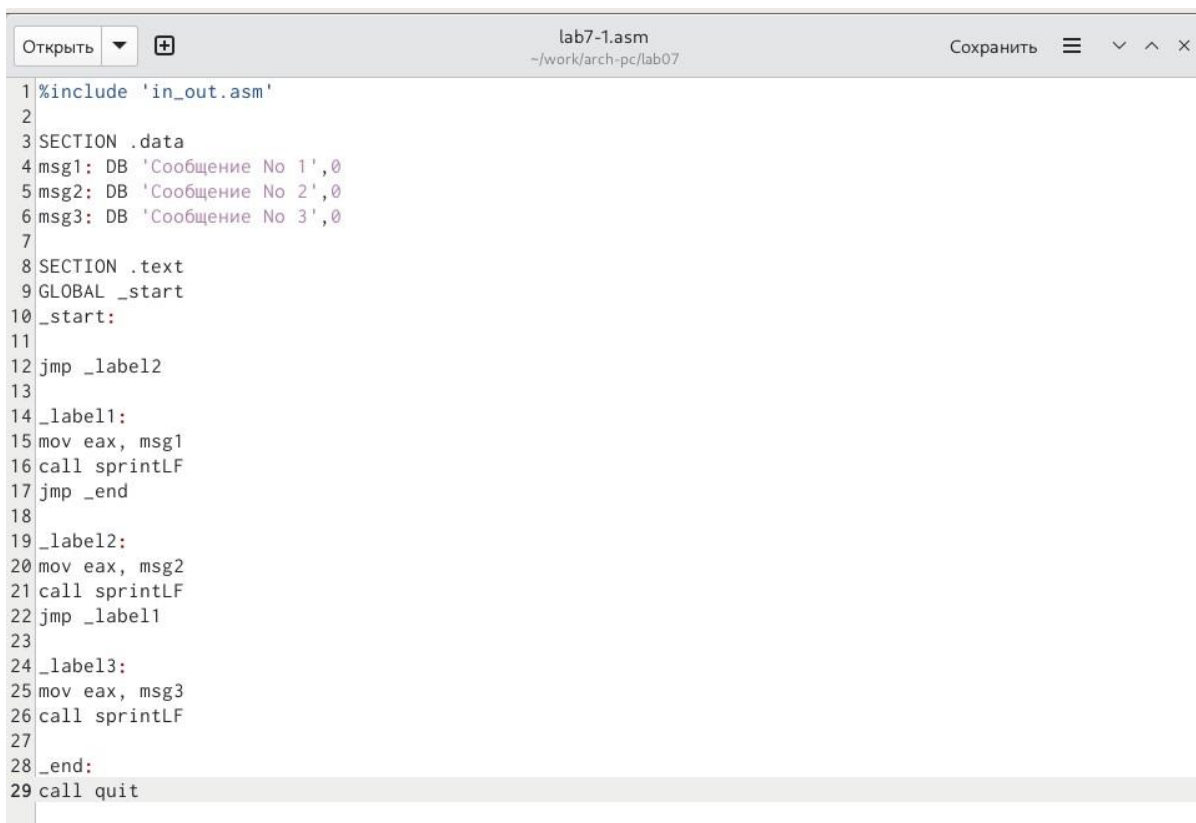
Рис. 2.2: рисунок

- Затем мы скомпилировали файл, создали исполняемый файл и запустили программу, все это после перемещения файла in\_out.asm в тот же каталог, где находится lab7-1.asm. (рис. [2.3])

```
[mabaptishta@fedora lab07]$ nasm -f elf lab7-1.asm
[mabaptishta@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[mabaptishta@fedora lab07]$ ./lab7-1
bash: ./lab7-1: Ficheiro ou pasta inexistente
[mabaptishta@fedora lab07]$ ./lab7-1
Сообщение No 2
Сообщение No 3
 falta de segmentação (nócleo desprotegido)
```

Рис. 2.3: рисунок

- После этого мы изменили код в листинге.(рис. [2.4])



```
lab7-1.asm
~/work/arch-pc/lab07

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

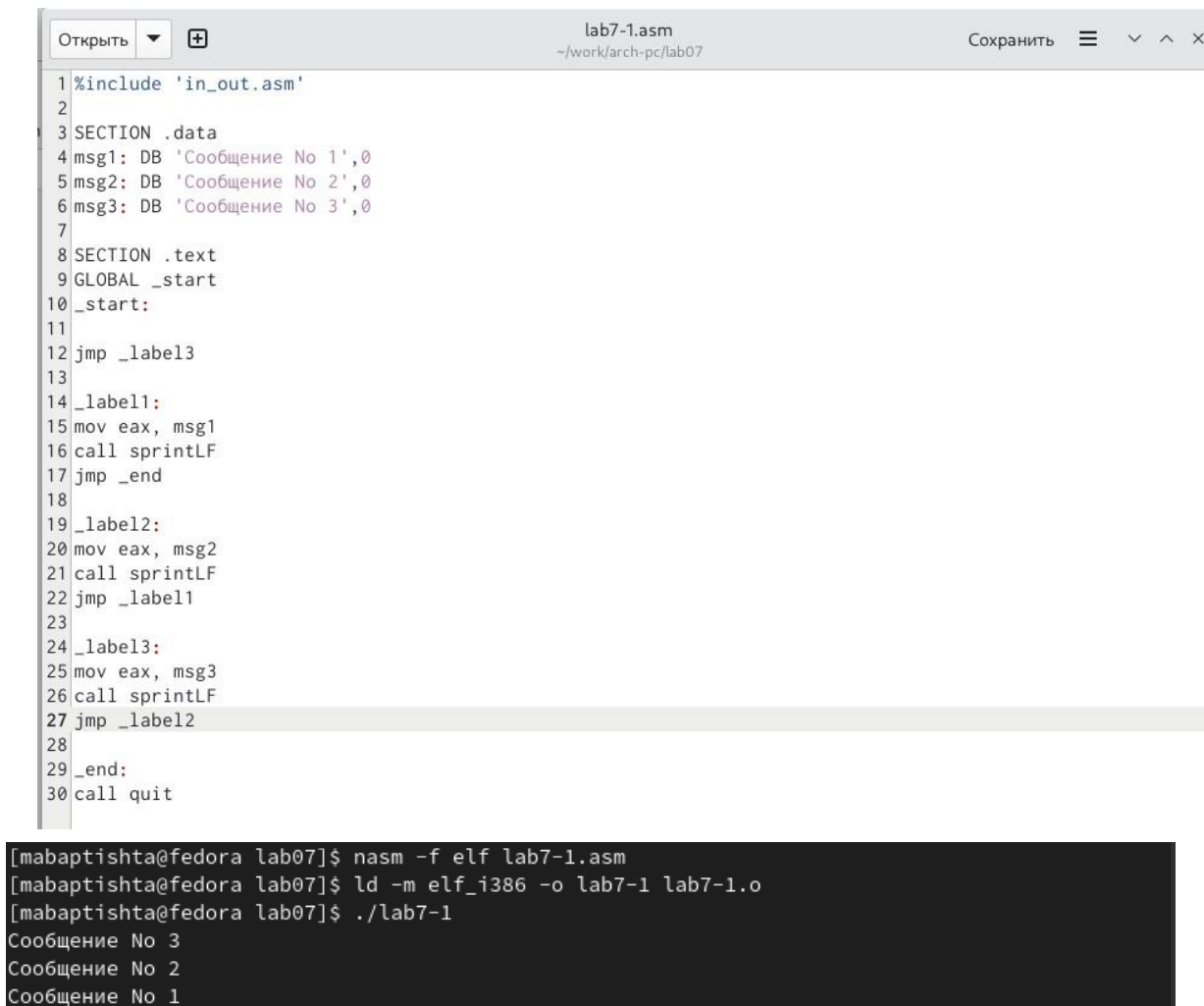
Рис. 2.4: рисунок

- Затем мы снова скомпилировали файл и создали исполняемый файл.(рис. [2.5])

```
[mabaptishta@fedora lab07]$ gcc -c lab7-1.c
[mabaptishta@fedora lab07]$ nasm -f elf lab7-1.asm
[mabaptishta@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[mabaptishta@fedora lab07]$ ./lab7-1
Сообщение No 2
Сообщение No 3
```

Рис. 2.5: рисунок

- Затем мы снова изменили код в листинге ,чтобы вывод программы был следующим: user@dk4n31:~\$ ./lab7-1 Сообщение No 3 Сообщение No 2 Сообщение No 1 user@dk4n31:~\$ (рис. [??])(рис. [??])



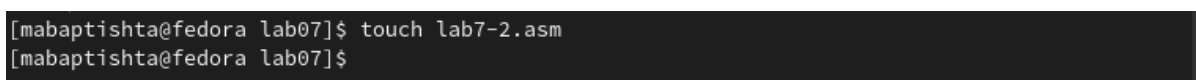
The image shows a code editor window titled 'lab7-1.asm' with the following assembly code:

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

Below the code editor is a terminal window showing the compilation and execution of the program:

```
[mabaptishta@fedora lab07]$ nasm -f elf lab7-1.asm
[mabaptishta@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[mabaptishta@fedora lab07]$ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
```

Рис. 2.6: рисунок



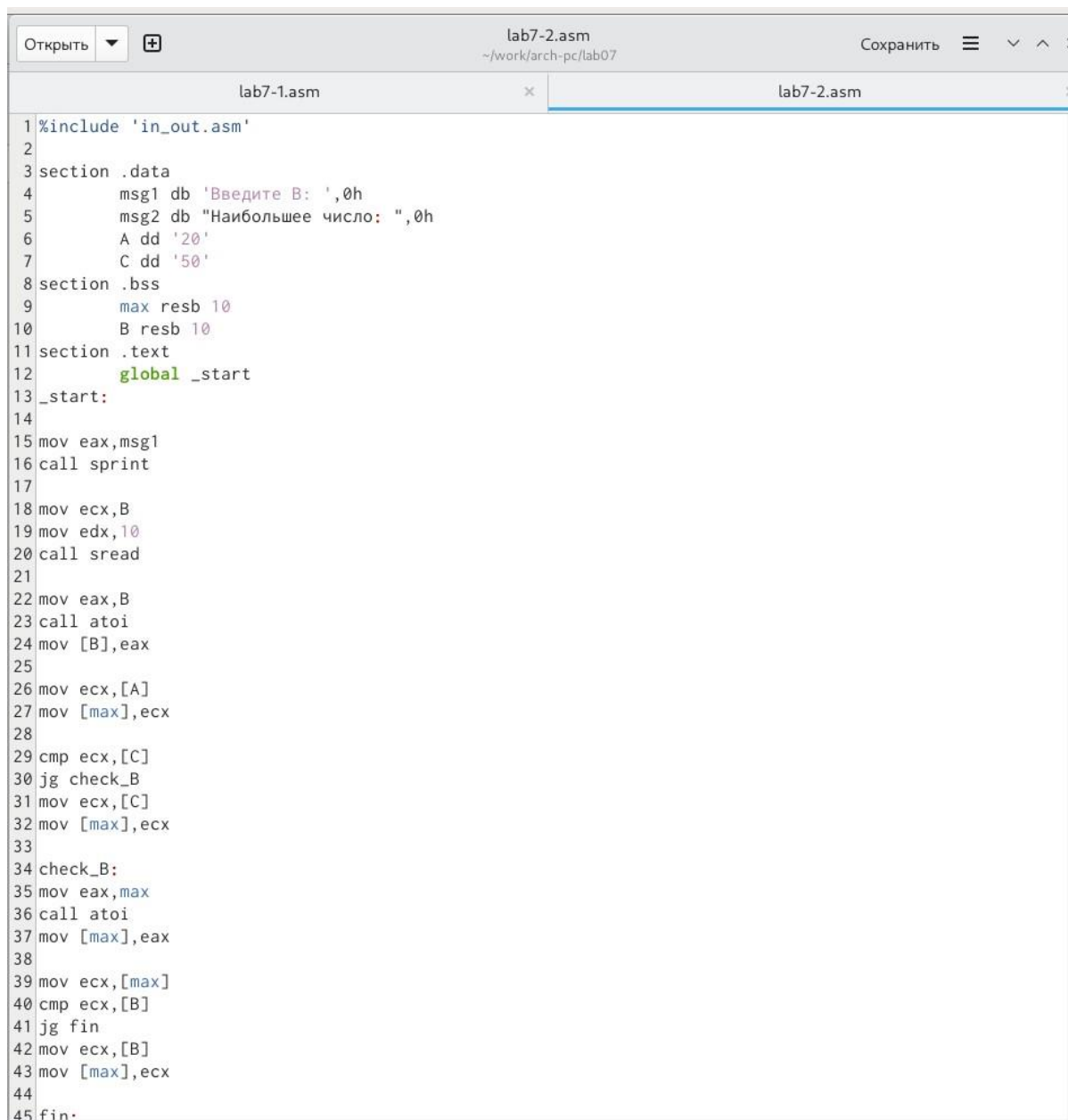
The image shows a terminal window with the following commands:

```
[mabaptishta@fedora lab07]$ touch lab7-2.asm
[mabaptishta@fedora lab07]$
```

- После этого мы создали файл lab7-2.asm, в который мы добавим код нашей следующей программы (рис. [2.6])

- После этого мы заполнили файл необходимым кодом для Программы, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А,В и С (рис. [2.7])





```
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 section .bss
9     max resb 10
10    B resb 10
11 section .text
12    global _start
13 _start:
14
15 mov eax,msg1
16 call sprint
17
18 mov ecx,B
19 mov edx,10
20 call sread
21
22 mov eax,B
23 call atoi
24 mov [B],eax
25
26 mov ecx,[A]
27 mov [max],ecx
28
29 cmp ecx,[C]
30 jg check_B
31 mov ecx,[C]
32 mov [max],ecx
33
34 check_B:
35 mov eax,max
36 call atoi
37 mov [max],eax
38
39 mov ecx,[max]
40 cmp ecx,[B]
41 jg fin
42 mov ecx,[B]
43 mov [max],ecx
44
45 fin
```

Рис. 2.7: рисунок

- мы скомпилировали файл,создали исполняемый файл и запустили его.(рис. [2.8])

```

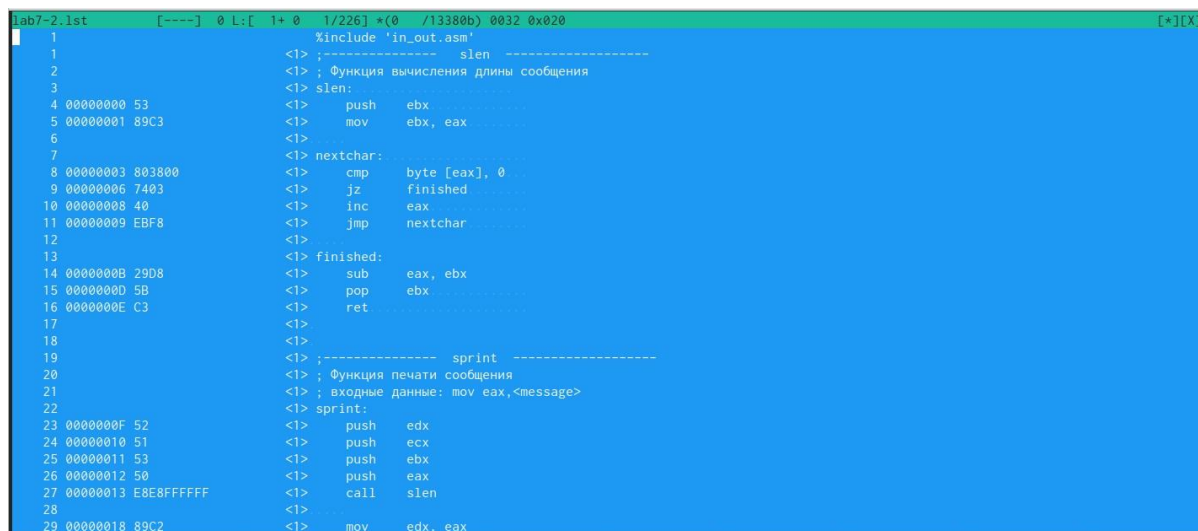
[mabaptishta@fedora lab07]$ touch lab7-2.asm
[mabaptishta@fedora lab07]$ nasm -f elf lab7-2.asm
[mabaptishta@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[mabaptishta@fedora lab07]$ ./lab7-2
Введите B: 10
Наибольшее число: 50
[mabaptishta@fedora lab07]$ ./lab7-2
Введите B: 52
Наибольшее число: 52
[mabaptishta@fedora lab07]$ ./lab7-2
Введите B: 60
Наибольшее число: 60
[mabaptishta@fedora lab07]$

```

Рис. 2.8: рисунок

## 2.1 Изучение структуры файлы листинга :

- Здесь и с помощью команды `nasm -f elf -l lab7-2.list lab7-2.asm` мы создали файл листинга файла `lab7-2.asm`, затем мы открыли файл с помощью `mcedit`.(рис. [2.9])



```

lab7-2.lst  [----]  0 L:[ 1+ 0 1/226] *(0 /13380b) 0032 0x020 [X]
1          %include 'in_out.asm'
2          <1> ;----- slen -----
3          <1> ; Функция вычисления длины сообщения
4          <1> slen:
5          <1>     push    ebx
6          <1>     mov     ebx, eax
7          <1>
8          <1> nextchar:
9          <1>     cmp     byte [eax], 0
10         <1>     jz      finished
11         <1>     inc     eax
12         <1>     jmp     nextchar
13         <1> finished:
14         <1>     sub     eax, ebx
15         <1>     pop     ebx
16         <1>     ret
17         <1>
18         <1>
19         <1> ;----- sprintf -----
20         <1> ; Функция печати сообщения
21         <1> ; входные данные: mov eax,<message>
22         <1> sprintf:
23         <1>     push    edx
24         <1>     push    ecx
25         <1>     push    ebx
26         <1>     push    eax
27         <1>     call    slen
28         <1>
29         <1>     mov     edx, eax

```

Рис. 2.9: рисунок

- мы выбрали эти три строки и пытаемся объяснить каждую из них.(рис. [2.10])

```

18 000000F2 B9[0A000000]    mov ecx,B
19 000000F7 BA0A000000    mov edx,10
20 000000FC E842FFFFFF    call sread

```

Рис. 2.10: рисунок

- Здесь мы переместили значение адреса переменной В в регистр ecx, после этого мы поместили значение 10 в регистр edx, который определяет размер переменной В с помощью подпрограммы sread и, наконец, мы вызвали подпрограмму sread
- мы открыли программный файл lab 7-2.asm и удалили один операнд в любой инструкции с двумя операндами. (рис. [2.11])

```

24 0000010B A3[0A000000]    mov [B],eax
25
26
27 00000116 890D[00000000]    mov [max],ecx
28
29 0000011C 3B0D[39000000]    cmp ecx,[C]

```

Рис. 2.11: рисунок

- В результате изменений был изменен файл листинга, в котором мы получили ошибку, объясняющую отсутствующий операнд, и файлы не были созданы.

## 2.2 Выводы по результатам выполнения заданий :

- Во время лабораторной работы мы узнали, как выполнять условные и безусловные переходы, как читать файл листинга.

## 3 Задание для самостоятельной работы

:

### **3.1 Написание программы нахождения наименьшей из 3 целочисленных переменных :**

- Мой код : (рис. [3.1])

```
1 %include 'in_out.asm'
2 section .data
3     msg1 db ' My values : 44,74,17',0h
4     msg2 db "The smallest number is : ",0h
5     A dd '44'
6     B dd '74'
7     C dd '17'
8 section .bss
9     min resb 10
10 section .text
11
12 global _start
13 _start:
14
15     mov eax,msg1
16     call sprintf
17
18     mov ecx,[A]
19     mov [min],ecx
20
21     cmp ecx,[B]
22     jl check_C
23
24     mov ecx,[B]
25     mov [min],ecx
26
27 check_C:
28
29     mov eax,min
30     call atoi
31     mov [min],eax
32
33     mov eax,C
34     call atoi
35     mov [C],eax
36
37     mov ecx,[min]
38     cmp ecx,[C]
39     jl fin
40
41     mov ecx,[C]
42     mov [min],ecx
43
44 fin:
```

Рис. 3.1: рисунок

- Вывод кода :(рис. [3.2])

```
[mabaptishta@fedora lab07]$ nasm -f elf lab7-3.asm
[mabaptishta@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[mabaptishta@fedora lab07]$ ./lab7-3
My values: 44,74,17
The smallest number is: 77
[mabaptishta@fedora lab07]$
```

Рис. 3.2: рисунок

## **3.2 Написание программы**

Мой код : (рис. [3.3])

```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msgA: DB 'Please enter a value for a : ', 0
5 msgX: DB 'Please enter a value for x : ', 0
6 msg3: DB 'The result is : ', 0
7 SECTION .bss
8 A: RESB 80
9 X: RESB 80
10
11 SECTION .text
12 GLOBAL _start
13
14 _start:
15 mov eax, msgA
16 call sprint
17 mov ecx, A
18 mov edx, 80
19 call sread
20 mov eax, A
21 call atoi
22 mov [A], eax
23
24 mov eax, msgX
25 call sprint
26 mov ecx, X
27 mov edx, 80
28 call sread
29 mov eax, X
30 call atoi
31 mov [X], eax
32
33 mov ebx, [A]
34 cmp ebx, 7
35 ja first
36 jmp second
37
38 first:
39 mov eax, [A]
40 sub eax, 7
41 call iprintLF
42 call quit
43
44 second:
45 mov eax, [X]
```

Рис. 3.3: рисунок

Вывод кода :(рис. [3.4])

```
[mabaptishta@fedora lab07]$ nasm -f elf lab7-4.asm
[mabaptishta@fedora lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o
[mabaptishta@fedora lab07]$ ./lab7-4
Please enter a value for a : 3
Please enter a value for x : 9
27
[mabaptishta@fedora lab07]$ ./lab7-4
Please enter a value for a : 6
Please enter a value for x : 4
24
[mabaptishta@fedora lab07]$
```

Рис. 3.4: рисунок



## 4 Выводы по результатам выполнения заданий :

- В этой части мы смогли применить наш полученный навык понятным способом, заставив программу вычислять конечное значение в зависимости от значений введенных переменных с использованием условных переходов.

## **5 Выводы, согласованные с целью работы :**

- В восьмой лаборатории мы в основном узнали, как использовать условные и безусловные переходы в NASM, как читать структуру файла листинга.

## **Список литературы**