

Шаблон отчёта по лабораторной работе №8

Мугари Абдеррахим , НКАбд-03-22

Содержание

1	Цель работы :	5
2	Выполнение лабораторной работы :	6
2.1	Реализация переходов в NASM :	6
2.2	Изучение структуры файлы листинга :	14
2.3	Выводы по результатам выполнения заданий :	18
3	Задание для самостоятельной работы :	19
3.1	Написание программы нахождения наименьшей из 3 целочислен- ных переменных :	19
3.2	Написание программы, которая выполняет математическую опе- рацию в зависимости от значения введенных переменных : . . .	21
3.3	Выводы по результатам выполнения заданий :	22
4	Выводы, согласованные с целью работы :	23

Список иллюстраций

2.1	Ресунок 1	6
2.2	Ресунок 2	7
2.3	Ресунок 3	8
2.4	Ресунок 4	9
2.5	Ресунок 5	10
2.6	Ресунок 6	11
2.7	Ресунок 7	12
2.8	Ресунок 8	12
2.9	Ресунок 9	13
2.10	Ресунок 10	14
2.11	Ресунок 11	15
2.12	Ресунок 12	15
2.13	Ресунок 13	17
3.1	Ресунок 14	20
3.2	Ресунок 15	21
3.3	Ресунок 16	21
3.4	Ресунок 17	22

Список таблиц

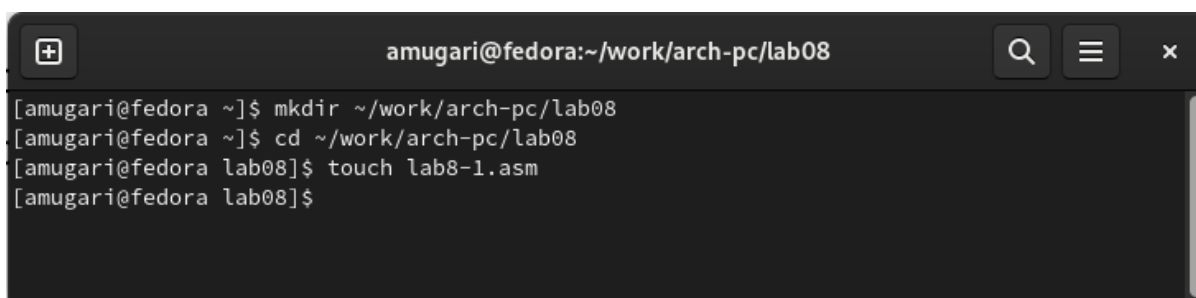
1 Цель работы :

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 :

2.1 Реализация переходов в NASM :

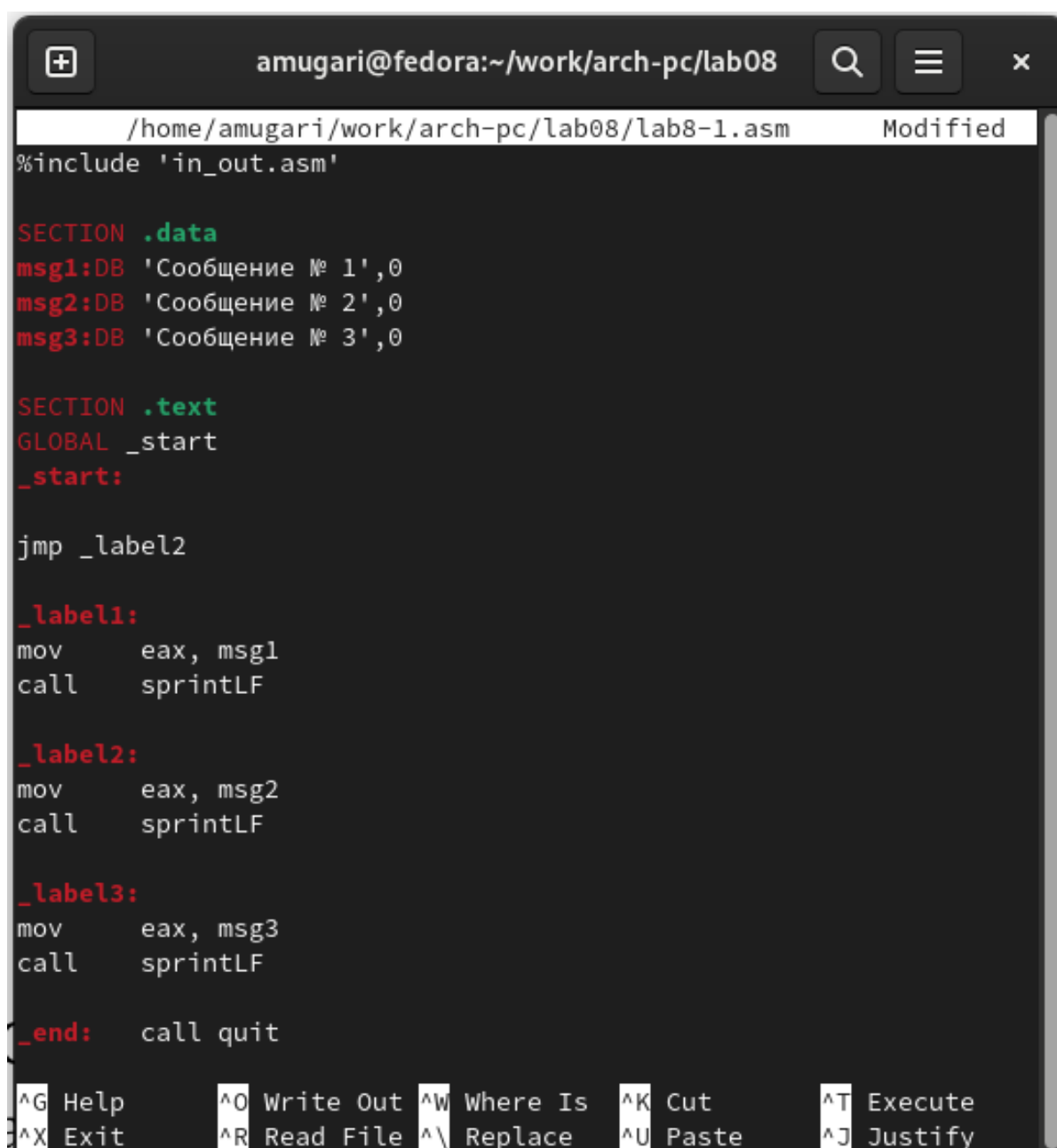
1. Здесь мы начали с создания, а затем переместились в восьмой каталог лаборатории “~/work/arch-pc/lab08”, после чего мы создали файл “**lab8-1.asm**”. (рис. 2.1)



```
amugari@fedora:~/work/arch-pc/lab08
[amugari@fedora ~]$ mkdir ~/work/arch-pc/lab08
[amugari@fedora ~]$ cd ~/work/arch-pc/lab08
[amugari@fedora lab08]$ touch lab8-1.asm
[amugari@fedora lab08]$
```

Рис. 2.1: Ресунок 1

2. После этого мы заполнили файл **.asm** кодом программы, отображающей значение регистра **eax**. (рис. 2.2)



```
amugari@fedora:~/work/arch-pc/lab08
/home/amugari/work/arch-pc/lab08/lab8-1.asm Modified
#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

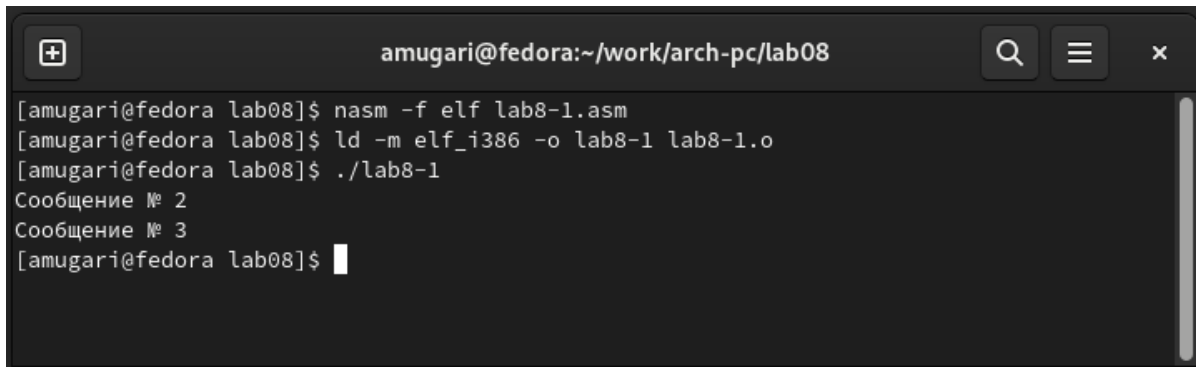
_label3:
mov     eax, msg3
call    sprintLF

_end:   call quit

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify
```

Рис. 2.2: Ресунок 2

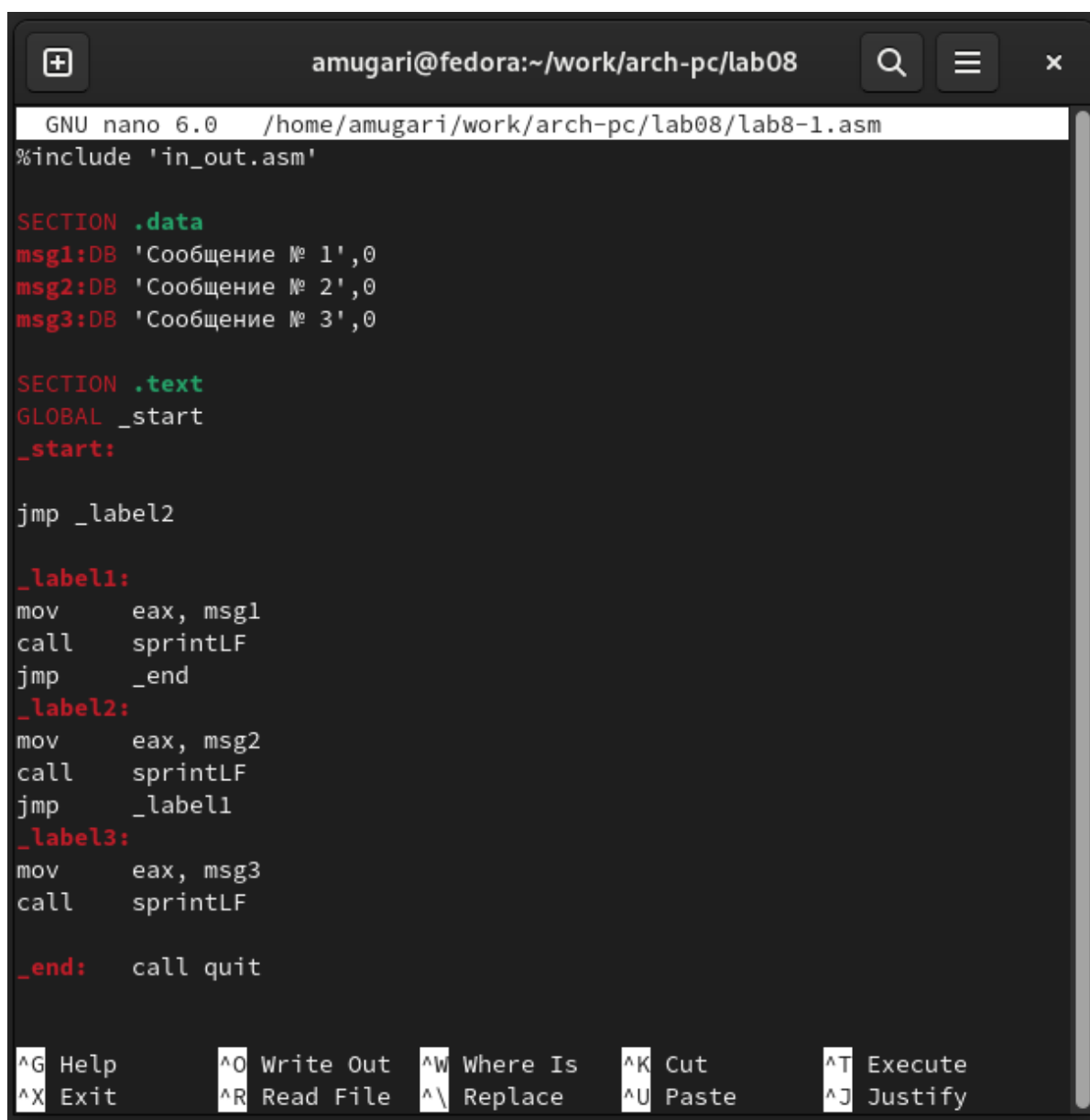
- Затем мы скомпилировали файл, создали исполняемый файл и запустили программу, все это после перемещения файла **in_out.asm** в тот же каталог, где находится **lab8-1.asm**. (рис. 2.3)

A terminal window with a dark background. The title bar shows the user 'amugari@fedora' and the directory '~/work/arch-pc/lab08'. The terminal contains the following text:

```
[amugari@fedora lab08]$ nasm -f elf lab8-1.asm
[amugari@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[amugari@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[amugari@fedora lab08]$
```

Рис. 2.3: Ресунок 3

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



```
GNU nano 6.0 /home/amugari/work/arch-pc/lab08/lab8-1.asm
#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

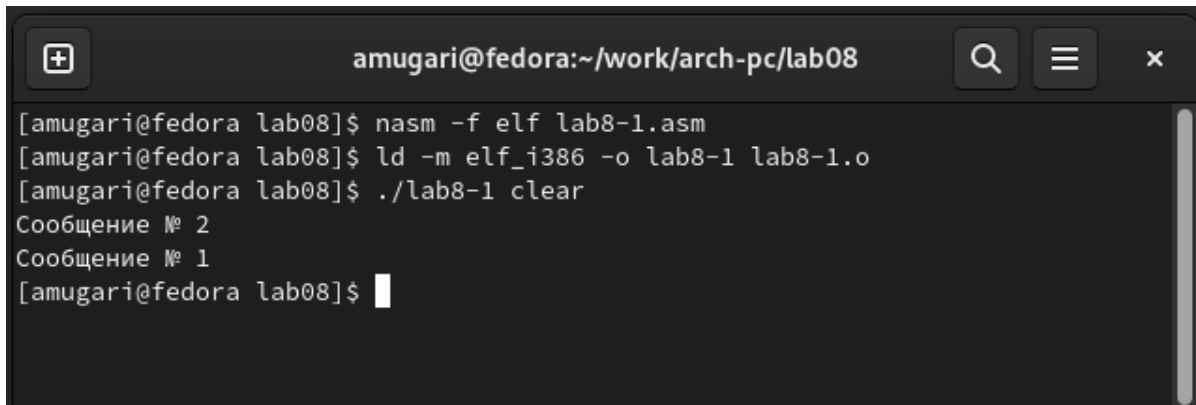
_label1:
mov     eax, msg1
call    sprintLF
jmp     _end
_label2:
mov     eax, msg2
call    sprintLF
jmp     _label1
_label3:
mov     eax, msg3
call    sprintLF

_end:   call quit

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify
```

Рис. 2.4: Ресунок 4

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

A terminal window with a dark background. The title bar shows the user 'amugari@fedora' and the directory '~/work/arch-pc/lab08'. The terminal contains the following text:

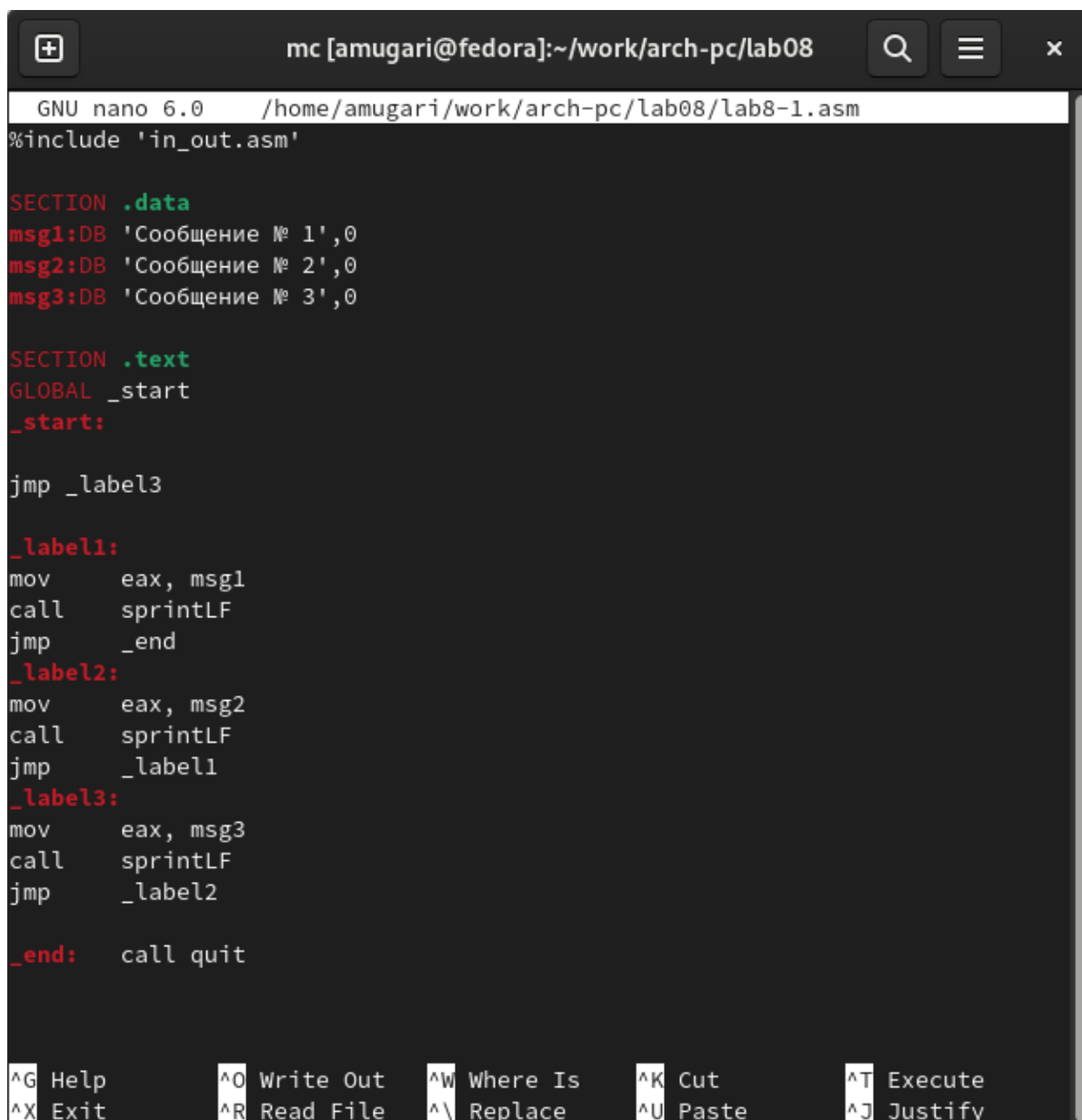
```
[amugari@fedora lab08]$ nasm -f elf lab8-1.asm
[amugari@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[amugari@fedora lab08]$ ./lab8-1 clear
Сообщение № 2
Сообщение № 1
[amugari@fedora lab08]$
```

Рис. 2.5: Ресунок 5

- Затем мы снова изменили код в листинге ,чтобы вывод программы был следующим:

```
user@dk4n31:~$ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
user@dk4n31:~$
```

(рис. 2.6) (рис. 2.7)



```
mc [amugari@fedora]:~/work/arch-pc/lab08
GNU nano 6.0 /home/amugari/work/arch-pc/lab08/lab8-1.asm
#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:

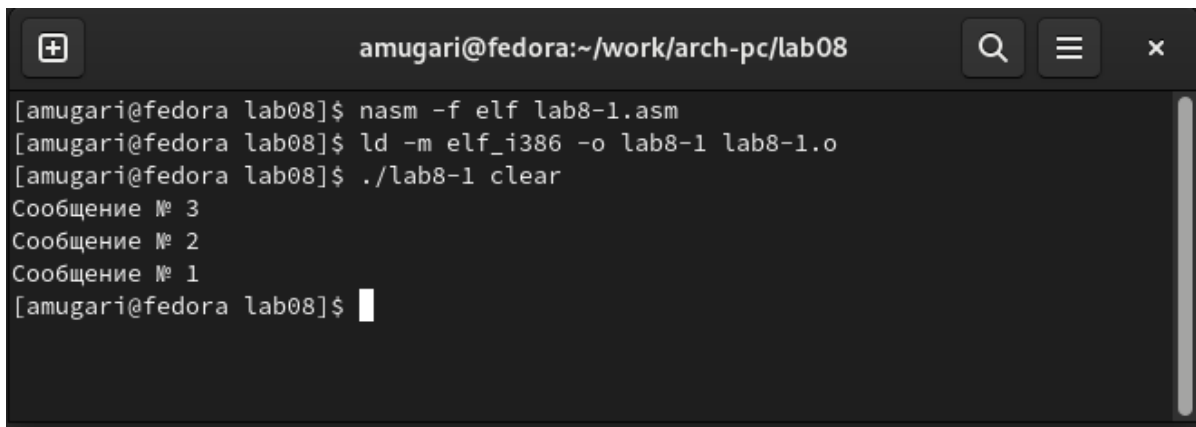
jmp _label3

_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

^G Help      ^O Write Out  ^W Where Is  ^K Cut       ^T Execute
^X Exit      ^R Read File  ^\ Replace   ^U Paste     ^J Justify
```

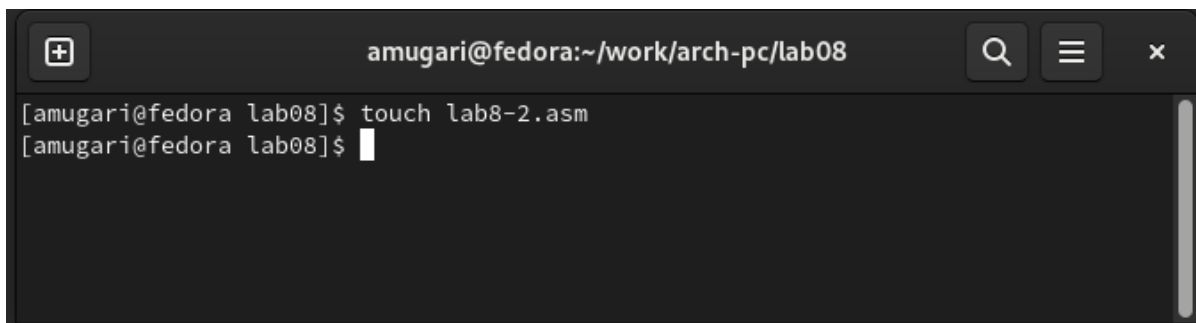
Рис. 2.6: Рисунок 6

A terminal window with a dark background. The title bar shows the user 'amugari' at host 'fedora' in the directory '~/work/arch-pc/lab08'. The terminal contains the following text:

```
[amugari@fedora lab08]$ nasm -f elf lab8-1.asm
[amugari@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[amugari@fedora lab08]$ ./lab8-1 clear
Сообщение № 3
Сообщение № 2
Сообщение № 1
[amugari@fedora lab08]$
```

Рис. 2.7: Рисунок 7

3. После этого мы создали файл **lab8-2.asm**, в который мы добавим код нашей следующей программы (рис. 2.8)

A terminal window with a dark background. The title bar shows the user 'amugari' at host 'fedora' in the directory '~/work/arch-pc/lab08'. The terminal contains the following text:

```
[amugari@fedora lab08]$ touch lab8-2.asm
[amugari@fedora lab08]$
```

Рис. 2.8: Рисунок 8

- После этого мы заполнили файл необходимым кодом для Программы, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C (рис. 2.9)

```
mc [amugari@fedora]:~/work/arch-pc/lab08
GNU nano 6.0 /home/amugari/work/arch-pc/lab08/lab8-2.asm
%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

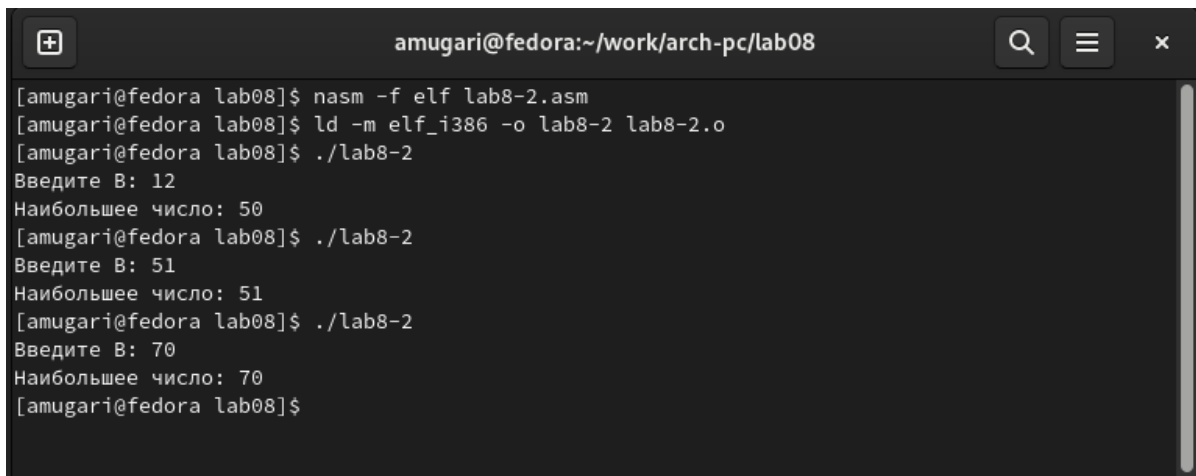
    cmp     ecx,[C]
    jg      check_B
    mov     ecx,[C]
    mov     [max],ecx
check_B:
    mov     eax,max
    call    atoi
    mov     [max],eax

    mov     ecx,[max]
    cmp     ecx,[B]
    jg      fin
    mov     ecx,[B]
    mov     [max],ecx
fin:
    mov     eax, msg2
    call    sprint
    mov     eax,[max]
    call    iprintLF
    call    quit

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Рис. 2.9: Ресунок 9

- мы скомпилировали файл, создали исполняемый файл и запустили его.
(рис. 2.10)



```
amugari@fedora:~/work/arch-pc/lab08
[amugari@fedora lab08]$ nasm -f elf lab8-2.asm
[amugari@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[amugari@fedora lab08]$ ./lab8-2
Введите В: 12
Наибольшее число: 50
[amugari@fedora lab08]$ ./lab8-2
Введите В: 51
Наибольшее число: 51
[amugari@fedora lab08]$ ./lab8-2
Введите В: 70
Наибольшее число: 70
[amugari@fedora lab08]$
```

Рис. 2.10: Ресунок 10

2.2 Изучение структуры файлы листинга :

1. Здесь и с помощью команды `nasm -f elf -l lab8-2.list lab8-2.asm` мы создали файл листинга файла **lab8-2.asm**, затем мы открыли файл с помощью **mcedit**.(рис. 2.11)

```

amugari@fedora:~/work/arch-pc/lab08 — mcedit lab8-2.lst
lab8-2.lst  [----] 0 L: [ 1+ 0 1/223] *(0 /13223b) 0032 0x020 [*] [X]
1      %include<----->'in_out.asm'
2      <1> ;----- slen -----
3      <1> ; Функция вычисления длины сообщения
4      <1> slen:.....
5      00000000 53      <1>      push     ebx.....
6      00000001 89C3    <1>      mov      ebx, eax.....
7      <1>.....
8      <1> nextchar:.....
9      00000003 803800  <1>      cmp      byte [eax], 0...
10     00000006 7403    <1>      jz       finished.....
11     00000008 40      <1>      inc      eax.....
12     00000009 EBF8    <1>      jmp      nextchar.....
13     <1>.....
14     <1> finished:
15     0000000B 29D8    <1>      sub      eax, ebx
16     0000000D 5B      <1>      pop      ebx.....
17     0000000E C3      <1>      ret.....
18     <1>.....
19     <1>.....
20     <1> ;----- sprint -----
21     <1> ; Функция печати сообщения
22     <1> ; входные данные: mov eax,<message>
23     <1> sprint:
24     0000000F 52      <1>      push     edx
25     00000010 51      <1>      push     ecx
26     00000011 53      <1>      push     ebx
27     00000012 50      <1>      push     eax
28     00000013 E8E8FFFFFF <1>      call     slen
29     <1>.....
30     00000018 89C2    <1>      mov      edx, eax
31     0000001A 58      <1>      pop      eax
32     <1>.....
33     0000001B 89C1    <1>      mov      ecx, eax
34     0000001D BB01000000 <1>      mov      ebx, 1
35     00000022 B804000000 <1>      mov      eax, 4
36     00000027 CD80    <1>      int      80h
37     <1>.....
38     00000029 5B      <1>      pop      ebx
39     0000002A 59      <1>      pop      ecx
40     0000002B 5A      <1>      pop      edx
41     0000002C C3      <1>      ret
42     <1>.....
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit

```

Рис. 2.11: Ресунок 11

2. мы выбрали эти три строки и пытаемся объяснить каждую из них

```

17.....
18 000000F2 B9[0A000000]      <----->mov <-->ecx,B
19 000000F7 BA0A000000      <----->mov<---->edx,10
20 000000FC E842FFFFFF      <----->call <-->sread

```

Рис. 2.12: Ресунок 12

- Здесь в 18-й строке мы переместили значение адреса переменной **B** в регистр **ecx**, после этого мы поместили значение **10** в регистре **edx**, который определяет размер переменной **B** с помощью подпрограммы **sread** и, наконец, мы вызвали подпрограмму **sread**
3. мы открыли программный файл lab 8-2.asm и удалили один операнд в любой инструкции с двумя операндами. Мы выбрали строку под номером 27.(рис. 2.13)


```
amugari@fedora:~/work/arch-pc/lab08 — mcedit lab8-2.lst
lab8-2.lst [B---] 0 L:[183+21 204/224] *(12229/13310b) 0032 0x020 [*] [X]
 8 00000000 <res Ah> <----->max resb 10
 9 0000000A <res Ah> <----->B<----->resb 10
10 <----->section>.text
11.....
12 global _start
13 _start:
14.....
15 000000E8 B8[00000000] <----->mov<---->eax,msg1
16 000000ED E81DFFFFFF <----->call <->sprint
17.....
18 000000F2 B9[0A000000] <----->mov <-->ecx,B
19 000000F7 BA0A000000 <----->mov<---->edx,10
20 000000FC E842FFFFFF <----->call <->sread
21.....
22 00000101 B8[0A000000] <----->mov<---->eax,B
23 00000106 E891FFFFFF <----->call<-->atoi
24 0000010B A3[0A000000] <----->mov<---->[B],eax
25 00000110 8B0D[35000000] <----->mov<---->ecx,[A]
26 00000116 890D[00000000] <----->mov<---->[max],ecx
27.....
28 <----->cmp<---->ecx
28 ***** error: invalid combination of opcode and operands
29 0000011C 7F0C <----->jg<----->check_B
30 0000011E 8B0D[39000000] <----->mov<---->ecx,[C]
31 00000124 890D[00000000] <----->mov<---->[max],ecx
32 check_B:
33 0000012A B8[00000000] <----->mov<---->eax,max
34 0000012F E868FFFFFF <----->call<-->atoi
35 00000134 A3[00000000] <----->mov<---->[max],eax
36.....
37 00000139 8B0D[00000000] <----->mov<---->ecx,[max]
38 0000013F 3B0D[0A000000] <----->cmp<---->ecx,[B]
39 00000145 7F0C <----->jg<----->fin
40 00000147 8B0D[0A000000] <----->mov<---->ecx,[B]
41 0000014D 890D[00000000] <----->mov<---->[max],ecx
42 fin:
43 00000153 B8[13000000] <----->mov<---->eax, msg2
44 00000158 E8B2FEFFFF <----->call <->sprint
45 0000015D A1[00000000] <----->mov<---->eax,[max]
46 00000162 E81FFFFFFF <----->call <->iprintLF
47 00000167 E86FFFFFFF <----->call <->quit
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit
```

Рис. 2.13: Ресунок 13

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

2.3 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 вариант : 13

- Мой код : (рис. 3.1)

```
amugari@fedora:~/work/arch-pc/lab08
GNU nano 6.0 /home/amugari/work/arch-pc/lab08/lab8-3.asm
#include "in_out.asm"
section .data
    msg1 db 'My values : 84,32,77',0h
    msg2 db "The smallest number is : ",0h
    A dd '84'
    B dd '32'
    C dd '77'
section .bss
    min resb 10
section .text

global _start
_start:

    mov     eax,msg1
    call    sprintLF

    mov     ecx,[A]
    mov     [min],ecx

    cmp     ecx,[B]
    jl      check_C

    mov     ecx,[B]
    mov     [min],ecx

check_C:
    mov     eax,min
    call    atoi
    mov     [min],eax

    mov     eax,C
    call    atoi
    mov     [C],eax

    mov     ecx,[min]
    cmp     ecx,[C]
    jl      fin

    mov     ecx,[C]
    mov     [min],ecx

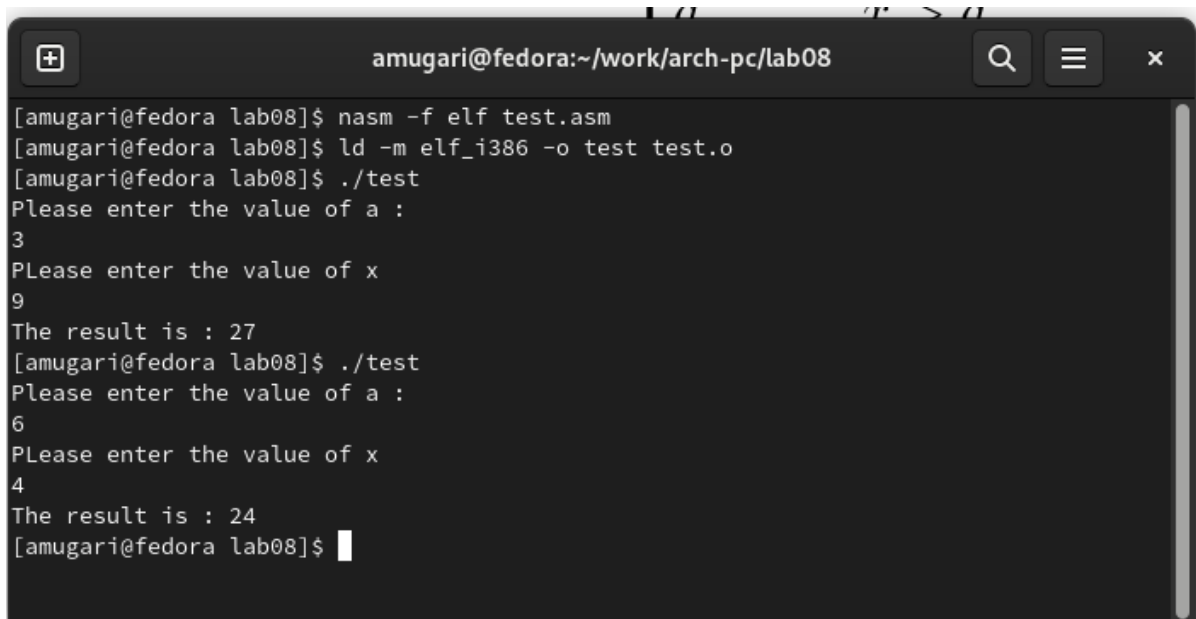
fin:
    mov     eax, msg2
    call    sprint
    mov     eax,[min]
    call    iprintLF
    call    quit

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^_ Replace   ^U Paste     ^J Justify   ^_/ Go To Line
```

Рис. 3.1: Ресунок 14

- Вывод кода :(рис. 3.2)

- Вывод кода :(рис. 3.4)



```
amugari@fedora:~/work/arch-pc/lab08
[amugari@fedora lab08]$ nasm -f elf test.asm
[amugari@fedora lab08]$ ld -m elf_i386 -o test test.o
[amugari@fedora lab08]$ ./test
Please enter the value of a :
3
Please enter the value of x
9
The result is : 27
[amugari@fedora lab08]$ ./test
Please enter the value of a :
6
Please enter the value of x
4
The result is : 24
[amugari@fedora lab08]$
```

Рис. 3.4: Ресунок 17

3.3 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.

4 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.