

# **Шаблон отчёта по лабораторной работе**

**8**

Сильвен Макс Грегор Филс б НКАбд-03-22

# 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 : . . . . .	13
2.2	Выводы по результатам выполнения заданий : . . . . .	15
<b>3</b>	<b>Задание для самостоятельной работы :</b>	<b>16</b>
3.1	Написание программы нахождения наименьшей из 3 целочисленных переменных : . . . . .	16
3.2	Написание программы, которая выполняет математическую операцию в зависимости от значения введенных переменных : . . .	18
<b>4</b>	<b>Выводы по результатам выполнения заданий :</b>	<b>21</b>
<b>5</b>	<b>Выводы, согласованные с целью работы :</b>	<b>22</b>
	<b>Список литературы</b>	<b>23</b>

## Список иллюстраций

2.1	Ресунок . . . . .	6
2.2	Ресунок . . . . .	7
2.3	Ресунок . . . . .	8
2.4	Ресунок . . . . .	9
2.5	Ресунок . . . . .	10
2.6	Ресунок . . . . .	11
2.7	Ресунок . . . . .	12
2.8	Ресунок . . . . .	13
2.9	Ресунок . . . . .	14
2.10	Ресунок . . . . .	14
2.11	Ресунок . . . . .	15
3.1	Ресунок . . . . .	17
3.2	Ресунок . . . . .	18
3.3	Ресунок . . . . .	19
3.4	Ресунок . . . . .	20

## Список таблиц

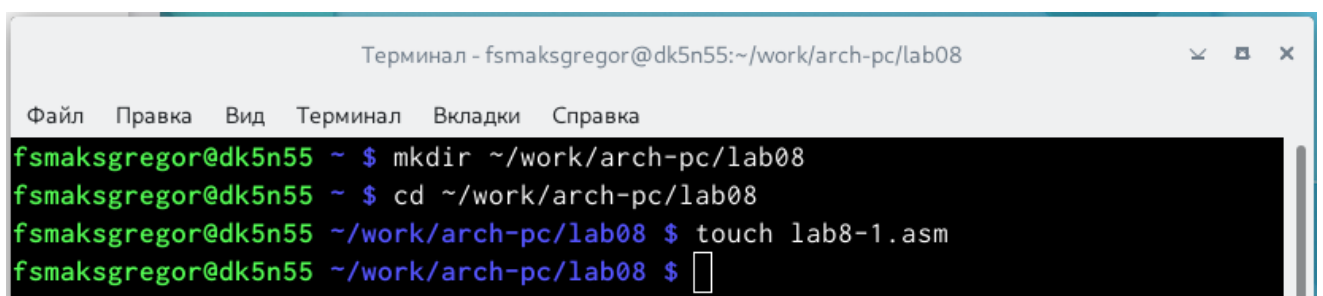
# 1 Цель работы

- 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 :

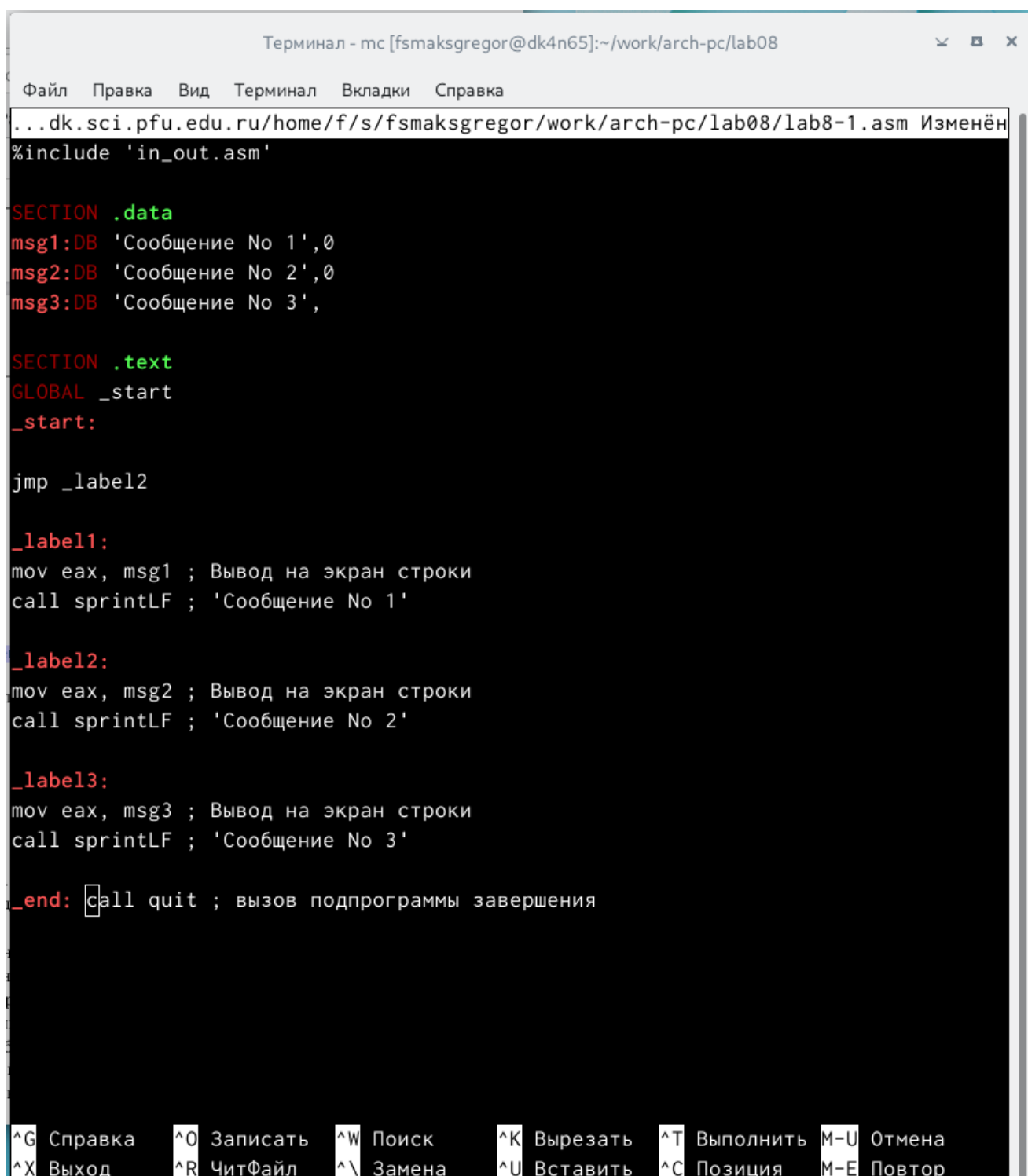
- Здесь мы начали с создания, а затем переместились в восьмой каталог лабо- ратории “~/work/arch-pc/lab08”, после чего мы создали файл “lab8-1.asm”.(рис. 2.1)



```
Терминал - fsmaksgregor@dk5n55:~/work/arch-pc/lab08
Файл  Правка  Вид  Терминал  Вкладки  Справка
fsmaksgregor@dk5n55 ~ $ mkdir ~/work/arch-pc/lab08
fsmaksgregor@dk5n55 ~ $ cd ~/work/arch-pc/lab08
fsmaksgregor@dk5n55 ~/work/arch-pc/lab08 $ touch lab8-1.asm
fsmaksgregor@dk5n55 ~/work/arch-pc/lab08 $
```

Рис. 2.1: Ресунок

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



Терминал - mc [fsmaksgregor@dk4n65]:~/work/arch-pc/lab08

Файл Правка Вид Терминал Вкладки Справка

...dk.sci.pfu.edu.ru/home/f/s/fsmaksgregor/work/arch-pc/lab08/lab8-1.asm Изменён

```
%include 'in_out.asm'

SECTION .data
msg1:DB 'Сообщение No 1',0
msg2:DB 'Сообщение No 2',0
msg3:DB 'Сообщение No 3',

SECTION .text
GLOBAL _start
_start:

jmp _label2

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

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

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintfLF ; 'Сообщение No 3'

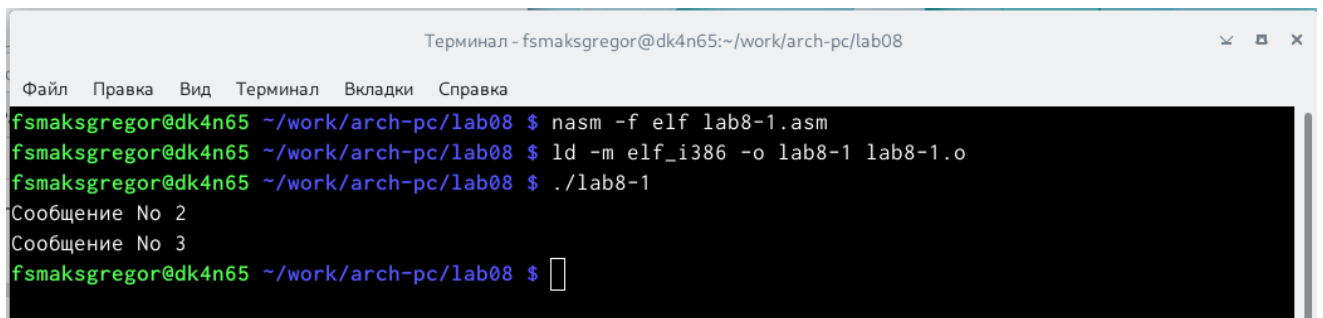
_end: call quit ; вызов подпрограммы завершения
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить M-U Отмена  
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^C Позиция M-E Повтор

Рис. 2.2: Ресунок

- 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)

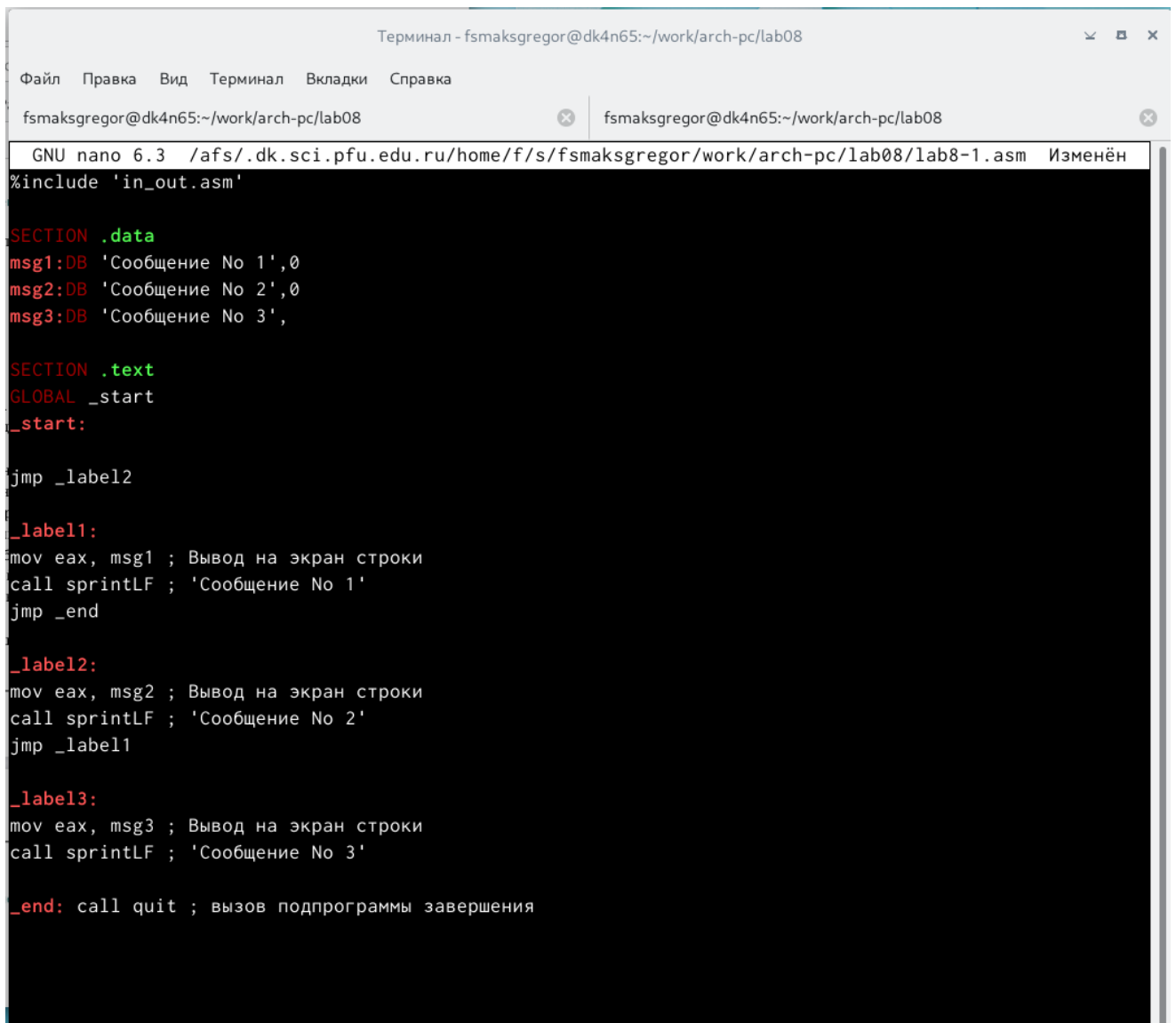


```
Терминал - fsmaksgregor@dk4n65:~/work/arch-pc/lab08
Файл  Правка  Вид  Терминал  Вкладки  Справка
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $ ./lab8-1
Сообщение No 2
Сообщение No 3
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $
```

Рис. 2.3: Ресунок

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





```
Терминал - fsmaksgregor@dk4n65:~/work/arch-pc/lab08
Файл  Правка  Вид  Терминал  Вкладки  Справка
fsmaksgregor@dk4n65:~/work/arch-pc/lab08  fsmaksgregor@dk4n65:~/work/arch-pc/lab08
GNU nano 6.3 /afs/.dk.sci.pfu.edu.ru/home/f/s/fsmaksgregor/work/arch-pc/lab08/lab8-1.asm  Изменён
#include 'in_out.asm'

SECTION .data
msg1:DB 'Сообщение No 1',0
msg2:DB 'Сообщение No 2',0
msg3:DB 'Сообщение No 3',

SECTION .text
GLOBAL _start
_start:

jmp _label2

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

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

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

_end: call quit ; вызов подпрограммы завершения
```

Рис. 2.4: Рисунок

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

```
mc [fsmaksgregor@dk4n65]:~/work/arch-pc/lab08
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $ ./lab8-1
Сообщение No 2
Сообщение No 1
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $
```

Рис. 2.5: Ресунок

- Затем мы снова изменили код в листинге ,чтобы вывод программы был следующим: user@dk4n31:~\$ ./lab8-1 Сообщение No 3 Сообщение No 2 Сообщение No 1 user@dk4n31:~\$ (рис. ??)(рис. ??)

```
mc [fsmaksgregor@dk4n65]:~/work/arch-pc/lab08
GNU nano 6.3 /afs/.dk.sci.pfu.edu.ru/home/f/s/fsmaksgregor/work/arch-pc/lab08/lab8-1.asm
%include 'in_out.asm'

SECTION .data
msg1:DB 'Сообщение No 1',0
msg2:DB 'Сообщение No 2',0
msg3:DB 'Сообщение No 3',

SECTION .text
GLOBAL _start
_start:

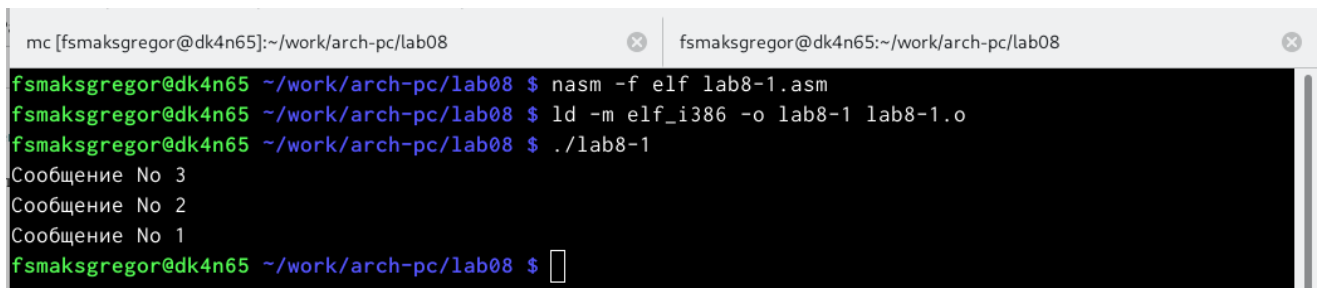
jmp _label3

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

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

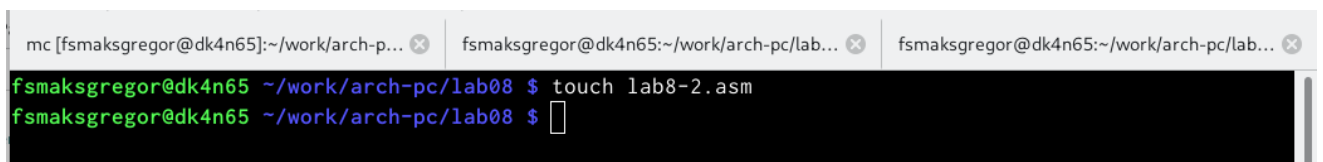
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
jmp _label2

_end: call quit ; вызов подпрограммы завершения
```



```
mc [fsmaksgregor@dk4n65]:~/work/arch-pc/lab08
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $ ./lab8-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $
```

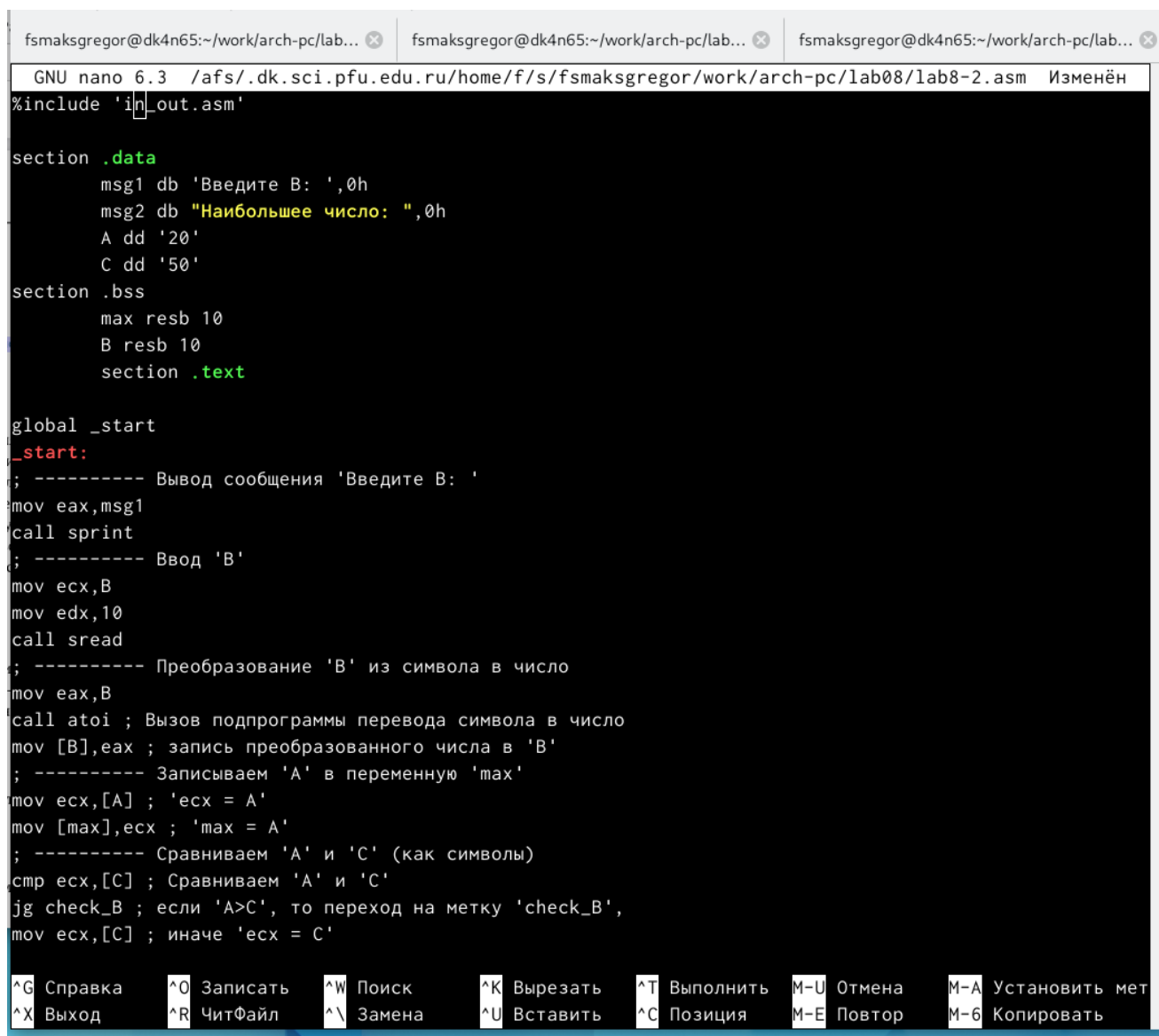
- После этого мы создали файл lab8-2.asm, в который мы добавим код нашей следующей программы (рис. 2.6)



```
mc [fsmaksgregor@dk4n65]:~/work/arch-p... fsmaksgregor@dk4n65:~/work/arch-pc/lab... fsmaksgregor@dk4n65:~/work/arch-pc/lab...
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $ touch lab8-2.asm
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $
```

Рис. 2.6: Ресунок

- 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. 2.7)



```
GNU nano 6.3 /afs/.dk.sci.pfu.edu.ru/home/f/s/fsmaksgregor/work/arch-pc/lab08/lab8-2.asm Изменён
%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:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
```

^G Справка    ^O Записать    ^W Поиск    ^K Вырезать    ^T Выполнить    M-U Отмена    M-A Установить мет  
^X Выход    ^R ЧитФайл    ^\ Замена    ^U Вставить    ^C Позиция    M-E Повтор    M-6 Копировать

Рис. 2.7: Ресунок

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

```
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $ ./lab8-2
Введите В: 10
Наибольшее число: 50
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $ ./lab8-2
Введите В: 51
Наибольшее число: 51
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $ ./lab8-2
Введите В: 60
Наибольшее число: 60
fsmaksgregor@dk4n65 ~/work/arch-pc/lab08 $ █
```

Рис. 2.8: Ресунок

## 2.1 Изучение структуры файлы листинга :

- 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. 2.9)

```

lab8-2.lst      [----] 68 L:[ 1+16 17/228] *(1143/14793b) 0010 0x00A [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 E8E8FFFF <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

```

1 Помощь 2 Сохранить 3 Блок 4 Замена 5 Копия 6 Переименовать 7 Поиск 8 Удалить 9 Меню MS 10 Выход

Рис. 2.9: Ресунок

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

```

19 000000F2 B9[0A000000]    mov ecx,B
20 000000F7 BA0A000000    mov edx,10
21 000000FC E842FFFFFF    call sread

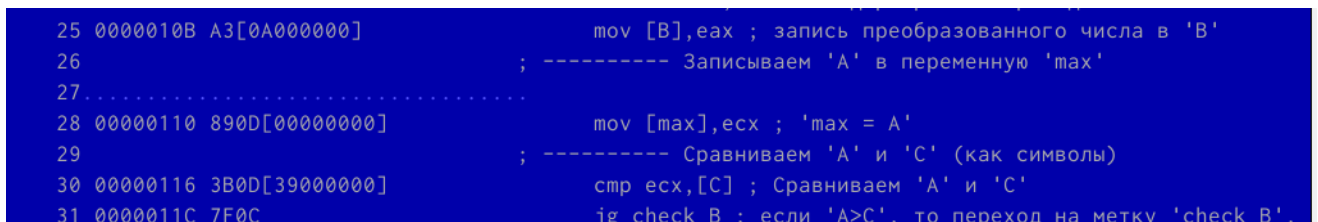
```

Рис. 2.10: Ресунок

- 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`

- 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. 2.11)



```
25 0000010B A3[0A000000]          mov [B],eax ; запись преобразованного числа в 'B'
26                               ; ----- Записываем 'A' в переменную 'max'
27.....
28 00000110 890D[00000000]          mov [max],ecx ; 'max = A'
29                               ; ----- Сравниваем 'A' и 'C' (как символы)
30 00000116 3B0D[39000000]          cmp ecx,[C] ; Сравниваем 'A' и 'C'
31 0000011C 7F0C                    jg check B ; если 'A>C', то переход на метку 'check B'.
```

Рис. 2.11: Ресунок

- 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 вариант : 16 - Мой код : (рис. 3.1)**



```
Файл  Правка  Вид  Терминал  Вкладки  Справка
mc [fsmaksgregor@dk6n50]:~/work/arch-pc/lab08  fsmaksgregor@dk6n50:~/work/arch-pc/lab08
test.asm  [----]  7  L:[  2+10  12/ 50]  *(239 / 751b)  0095  0x05F  [*][X]
section .data
    msg1 db ' My values : 44,74,17',0h
    msg2 db "The smallest number is : ",0h
    A dd '44'
    B dd '74'
    C dd '17'
section .bss
    min resb 10
section .text

global _start
_start:

    mov  eax,msg1
    call sprintf

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

    cmp  ecx,[B]
    jl  check_C

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

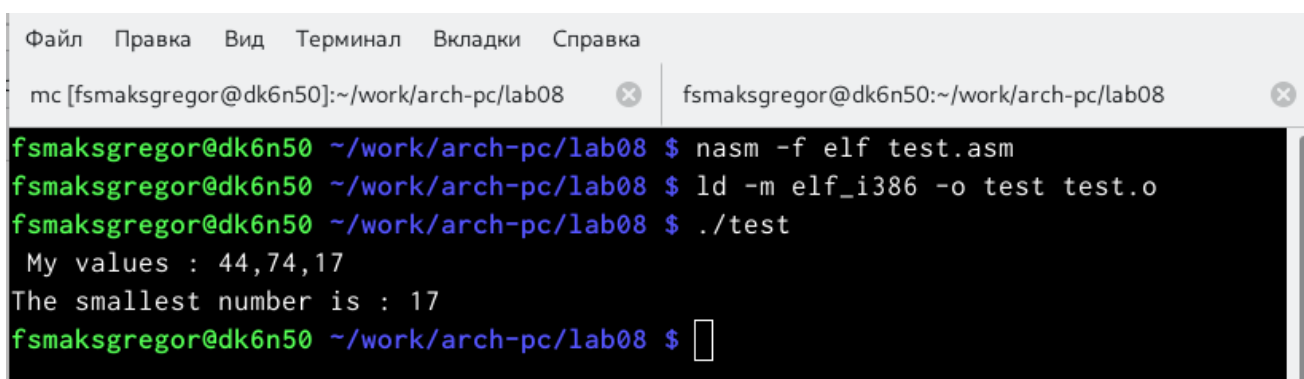
check_C:

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

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

Рис. 3.1: Рисунок

- Вывод кода :(рис. 3.2)



The screenshot shows a terminal window with a menu bar at the top containing 'Файл', 'Правка', 'Вид', 'Терминал', 'Вкладки', and 'Справка'. Below the menu bar, there are two tabs: 'mc [fsmaksgregor@dk6n50]:~/work/arch-pc/lab08' and 'fsmaksgregor@dk6n50:~/work/arch-pc/lab08'. The terminal content shows the following commands and output:

```
fsmaksgregor@dk6n50 ~/work/arch-pc/lab08 $ nasm -f elf test.asm
fsmaksgregor@dk6n50 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o test test.o
fsmaksgregor@dk6n50 ~/work/arch-pc/lab08 $ ./test
My values : 44,74,17
The smallest number is : 17
fsmaksgregor@dk6n50 ~/work/arch-pc/lab08 $
```

Рис. 3.2: Ресунок

## 3.2 Написание программы, которая выполняет математическую операцию в зависимости от значения введенных переменных :

$\{x - 7, x \geq 7\} \{xxx, x < 7$

Мой код : (рис. 3.3)

```
Файл  Правка  Вид  Терминал  Вкладки  Справка
mc [fsmaksgregor@dk6n50]:~/work/arch-pc/lab08  fsmaksgregor@dk6n50:~/work/arch-pc/lab08
test-2.asm  [----]  0 L:[ 1+ 9 10/ 50] *(192 / 576b) 0010 0x00A  [*][X]
#include 'in_out.asm'

SECTION .data.
msgA: DB 'Please enter a value for a : ', 0
msgX: DB 'Please enter a value for x : ', 0
msg3: DB 'The result is : ', 0
SECTION .bss
A: RESB 80
X: RESB 80
SECTION .text
GLOBAL _start

_start:
mov eax, msgA
call sprint
mov ecx, A
mov edx, 80
call sread
mov eax, A
call atoi
mov [A], eax

mov eax, msgX
call sprint
mov ecx, X
mov edx, 80
call sread
mov eax, X
call atoi
```

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9МенюМС10Выход

Рис. 3.3: Рисунок

Вывод кода :(рис. 3.4)

```
fsmaksgregor@dk6n50 ~/work/arch-pc/lab08 $ nasm -f elf test-2.asm
fsmaksgregor@dk6n50 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o test-2 test-2.o
fsmaksgregor@dk6n50 ~/work/arch-pc/lab08 $ ./test-2
Please enter a value for a : 3
Please enter a value for x : 9
27
fsmaksgregor@dk6n50 ~/work/arch-pc/lab08 $ ./test-2
Please enter a value for a : 6
Please enter a value for x : 4
24
fsmaksgregor@dk6n50 ~/work/arch-pc/lab08 $
```

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**