

Отчёт по лабораторной работе №7

**Команды безусловного и условного переходов в Nasm.
Программирование ветвлений.**

Власов Артем Сергеевич

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
3.1	Реализация переходов в NASM	6
3.2	Изучение структуры файлы листинга	10
3.3	Задание для самостоятельной работы	12
4	Выводы	16

Список иллюстраций

3.1	Создаем каталог с помощью команды <code>mkdir</code> и файл с помощью команды <code>touch</code>	6
3.2	Заполняем файл	6
3.3	Запускаем файл и смотрим на его работу	7
3.4	Изменяем файл	7
3.5	Запускаем файл и смотрим на его работу	7
3.6	Редактируем файл	8
3.7	Проверяем, сошелся ли наш вывод с данным в условии выводом .	8
3.8	Создаем файл командой <code>touch</code>	8
3.9	Заполняем файл	9
3.10	Смотрим на работу программ	9
3.11	Создаем файл листинга	10
3.12	Изучаем файл	10
3.13	Удаляем операндум из файла	11
3.14	Транслируем файл	11
3.15	Изучаем файл с ошибкой	12
3.16	Создаем файл командой <code>touch</code>	12
3.17	Пишем программу	13
3.18	Смотрим на работу программы(всё верно)	13
3.19	Создаем файл командой <code>touch</code>	14
3.20	Пишем программу	14
3.21	Проверяем работу программы	15
3.22	Проверяем работу программы	15

1 Цель работы

Освоить условного и безусловного перехода. Ознакомиться с назначением и структурой файла листинга.

2 Задание

Написать программы для решения системы выражений.

3 Выполнение лабораторной работы

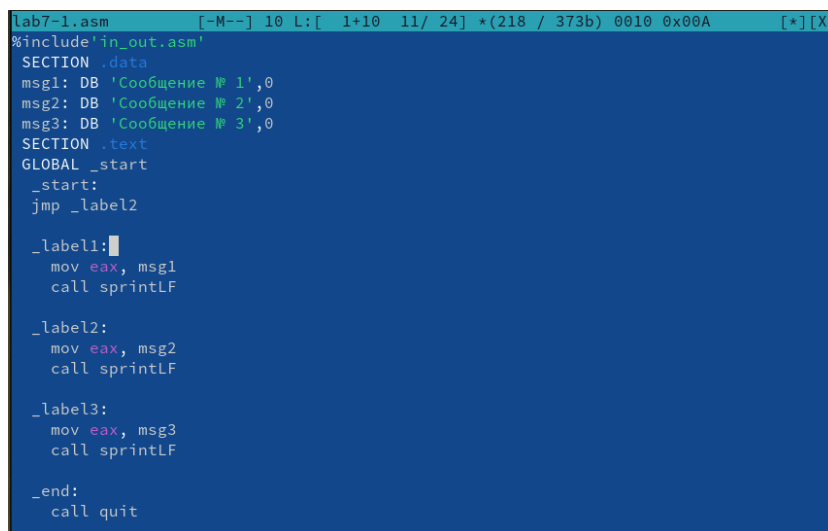
3.1 Реализация переходов в NASM

Создаем каталог для программ ЛБ7, и в нем создаем файл (рис. fig. 3.1).

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

Рис. 3.1: Создаем каталог с помощью команды `mkdir` и файл с помощью команды `touch`

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 7.1 (рис. fig. 3.2).



```
lab7-1.asm [-M--] 10 L: [ 1+10 11/ 24] *(218 / 373b) 0010 0x00A [*][X]
%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 sprintf

_label2:
mov eax, msg2
call sprintf

_label3:
mov eax, msg3
call sprintf

_end:
call quit
```

Рис. 3.2: Заполняем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.3).

```

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

```

Рис. 3.3: Запускаем файл и смотрим на его работу

Снова открываем файл для редактирования и изменяем его в соответствии с листингом 7.2 (рис. fig. 3.4).

```

#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 printf
jmp _end

_label2:
mov eax, msg2
call printf
jmp _label1

_label3:
mov eax, msg3
call printf

_end:
call quit

```

Рис. 3.4: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.5).

```

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

```

Рис. 3.5: Запускаем файл и смотрим на его работу

Снова открываем файл для редактирования и изменяем его, чтобы произошел данный вывод (рис. fig. 3.6).

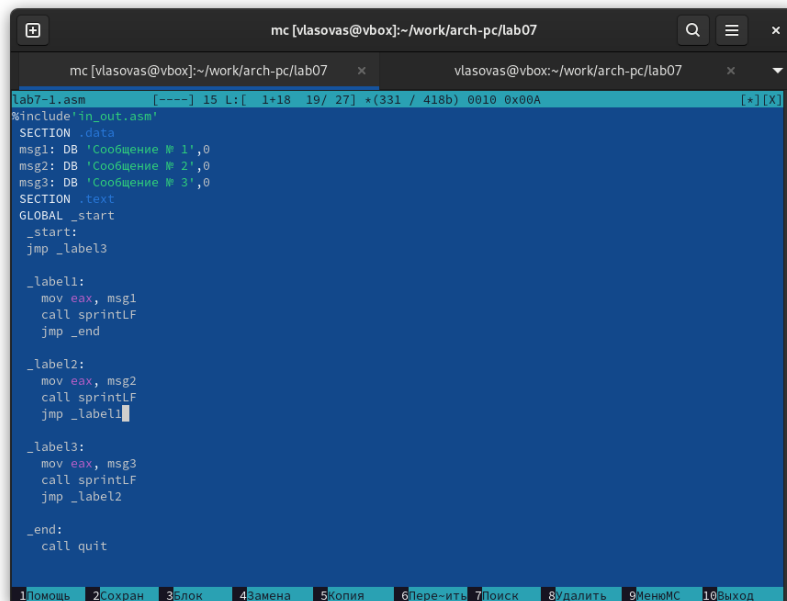


Рис. 3.6: Редактируем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.7).

```

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

Рис. 3.7: Проверяем, сошелся ли наш вывод с данным в условии выводом

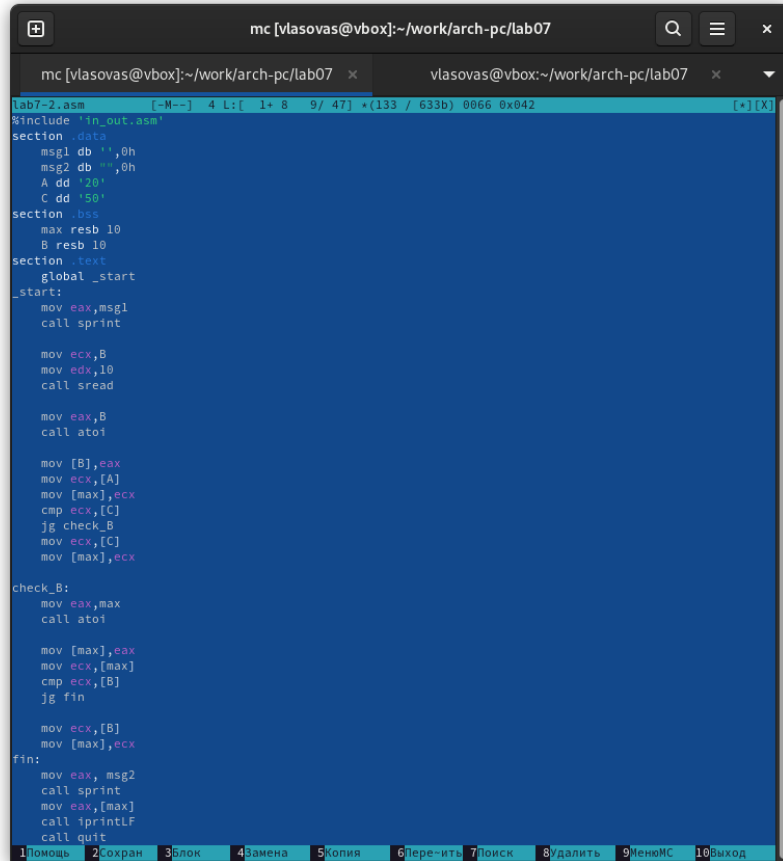
Создаем новый файл (рис. fig. 3.8).

```

vlasovas@vbox: ~/work/arch-pc/lab07$ touch lab7-2.asm
vlasovas@vbox: ~/work/arch-pc/lab07$
  
```

Рис. 3.8: Создаем файл командой touch

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 7.3 (рис. fig. 3.9).



```
mc [vlasovas@vbox]:~/work/arch-pc/lab07
lab7-2.asm [-M--] 4 L: [ 1+ 8 9/ 47] *(133 / 633b) 0066 0x042 [*] [X]
#include 'in_out.asm'
section .data
msg1 db ' ',0h
msg2 db ' ',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

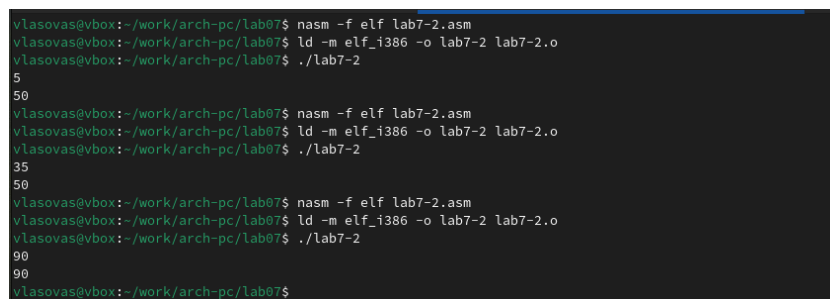
mov ecx,[B]
mov [max],ecx

fin:
mov eax, msg2
call sprint
mov eax,[max]
call printf
call quit
```

1.Помощь 2.Сохран 3.Блок 4.Замена 5.Копия 6.Пере-ить 7.Поиск 8.Удалить 9.МенюМС 10.Выход

Рис. 3.9: Заполняем файл

Создаем исполняемый файл и проверяем его работу, вводя разные значения В (рис. fig. 3.10).



```
vlasovas@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
vlasovas@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
vlasovas@vbox:~/work/arch-pc/lab07$ ./lab7-2
5
50
vlasovas@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
vlasovas@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
vlasovas@vbox:~/work/arch-pc/lab07$ ./lab7-2
35
50
vlasovas@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
vlasovas@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
vlasovas@vbox:~/work/arch-pc/lab07$ ./lab7-2
90
90
vlasovas@vbox:~/work/arch-pc/lab07$
```

Рис. 3.10: Смотрим на работу программ

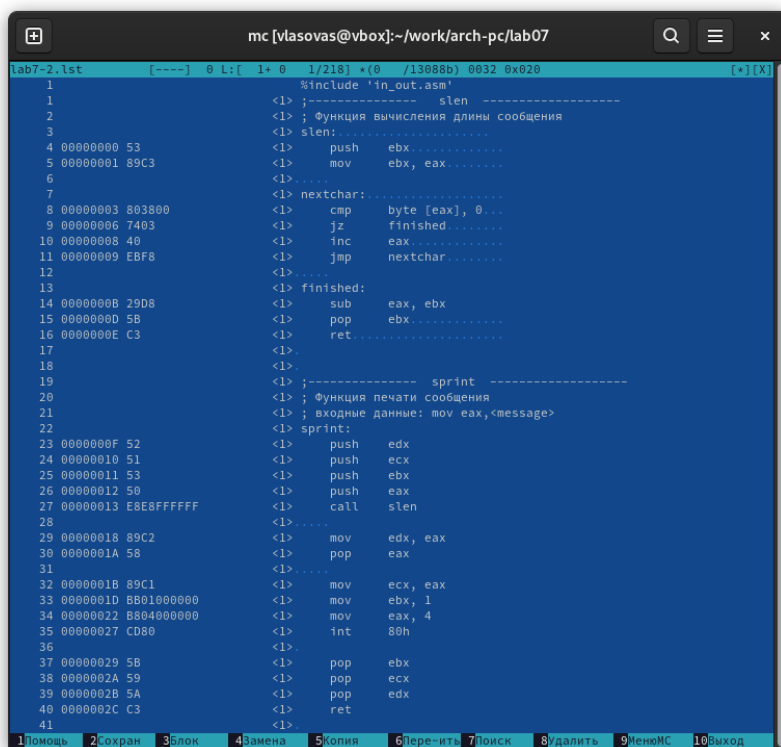
3.2 Изучение структуры файлы листинга

Создаем файл листинга для программы lab7-2.asm (рис. fig. 3.11).

```
vlasovas@vbox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
vlasovas@vbox:~/work/arch-pc/lab07$
```

Рис. 3.11: Создаем файл листинга

Открываем файл листинга с помощью команды `mcedit` и изучаем его (рис. fig. 3.12).



```
lab7-2.lst  [----]  0 L:  1+ 0  1/218]  *(0  /13088b)  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> finished:-----
14         0000000B 29D8        <1> sub     eax, ebx,-----
15         0000000D 5B          <1> pop     ebx,-----
16         0000000E C3          <1> ret,-----
17         <1>
18         <1>
19         <1> ;----- sprint -----
20         <1> ; Функция печати сообщения
21         <1> ; входные данные: mov eax, <message>
22         <1> sprint:-----
23         0000000F 52          <1> push    edx,-----
24         00000010 51          <1> push    ecx,-----
25         00000011 53          <1> push    ebx,-----
26         00000012 50          <1> push    eax,-----
27         00000013 E8E8FFFF    <1> call    slen,-----
28         <1>
29         00000018 89C2        <1> mov     edx, eax,-----
30         0000001A 58          <1> pop     eax,-----
31         <1>
32         0000001B 89C1        <1> mov     ecx, eax,-----
33         0000001D B801000000  <1> mov     ebx, 1,-----
34         00000022 B804000000  <1> mov     eax, 4,-----
35         00000027 CD80        <1> int     80h,-----
36         <1>
37         00000029 5B          <1> pop     ebx,-----
38         0000002A 59          <1> pop     ecx,-----
39         0000002B 5A          <1> pop     edx,-----
40         0000002C C3          <1> ret,-----
41         <1>
```

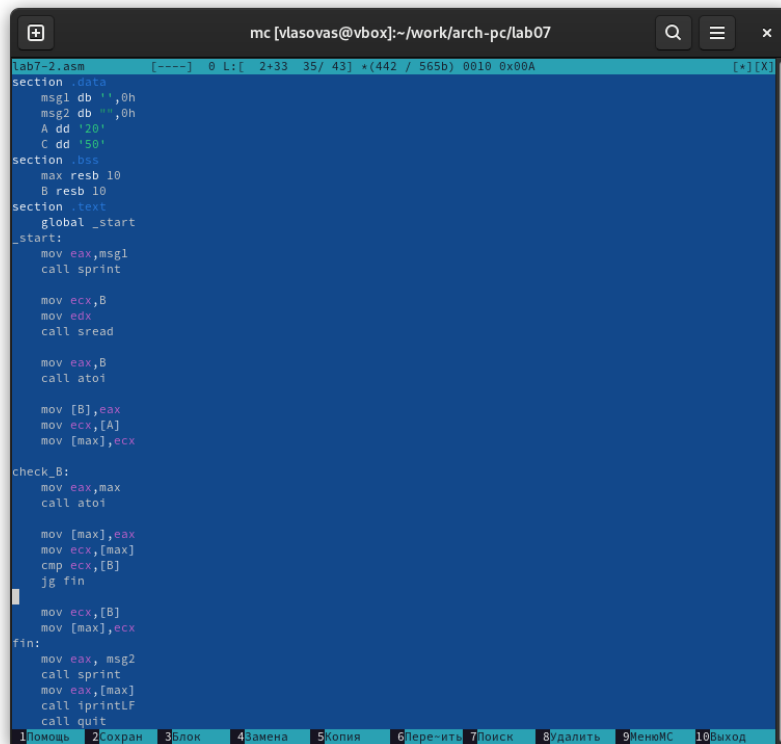
Рис. 3.12: Изучаем файл

Строка 33: 0000001D-адрес в сегменте кода, B801000000-машинный код, `mov ebx,1`-присвоение переменной `ebx` значения 1.

Строка 34: 00000022-адрес в сегменте кода, B804000000-машинный код, `mov eax,4`-присвоение переменной `eax` значения 4.

Строка 35 00000027-адрес в сегменте кода, CD80-машинный код, `int 80h`-вызов ядра.

Открываем файл и удаляем один операндум (рис. fig. 3.13).



```
lab7-2.asm [----] 0 L: [ 2+33 35/ 43] *(442 / 565b) 0010 0x00A [*] [X]
section .data
msg1 db "",0h
msg2 db "",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
call sread

mov eax,B
call atoi

mov [B],eax
mov ecx,[A]
mov [max],ecx

check_B:
mov eax,max
call atoi

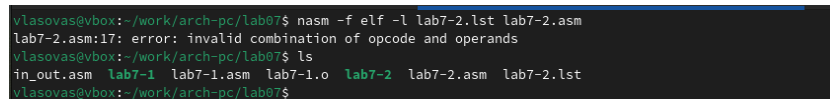
mov [max],eax
mov ecx,[max]
cmp ecx,[B]
jg fin

mov ecx,[B]
mov [max],ecx

fin:
mov eax, msg2
call sprint
mov eax,[max]
call fprintf
call quit
```

Рис. 3.13: Удаляем операндум из файла

Транслируем с получением файла листинга (рис. fig. 3.14).



```
vlasovas@vbox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:17: error: invalid combination of opcode and operands
vlasovas@vbox:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.lst
vlasovas@vbox:~/work/arch-pc/lab07$
```

Рис. 3.14: Транслируем файл

При трансляции файла, выдается ошибка, но создаются исполнительный файл lab7-2 и lab7-2.lst

Снова открываем файл листинга и изучаем его (рис. fig. 3.15).

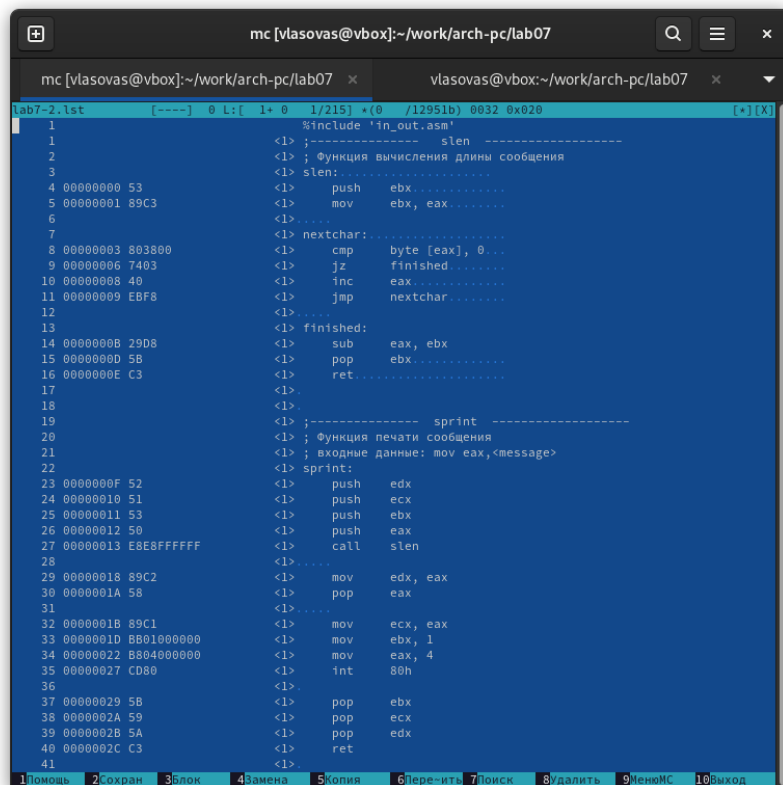


Рис. 3.15: Изучаем файл с ошибкой

3.3 Задание для самостоятельной работы

ВАРИАНТ-2

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных \boxed{x} , \boxed{y} и \boxed{z} . Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу.

Создаем новый файл (рис. fig. 3.16).

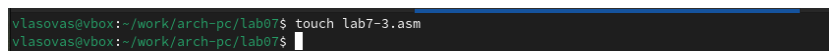


Рис. 3.16: Создаем файл командой touch

Открываем его и пишем программу, которая выберет наименьшее число из трех(2 числа уже в программе, 3е вводится из консоли) (рис. fig. 3.17).

```

mc [vlasovas@vbox]:~/work/arch-pc/lab07
lab7-3.asm [----] 14 L: 1+22 23/ 41] * (406 / 674b) 0010 0x00A [+] [X]
#include "in_out.asm"
section .data
    msg1 db "Введите B: ",0h
    msg2 db "Наименьшее число: ",0h
    A dd '82'
    C dd '61'
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]
    jl check_B
    mov ecx,[C]
    mov [min],ecx
check_B:
    mov eax,min
    call atoi
    mov [min],eax
    mov ecx,[min]
    cmp ecx,[B]
    jl fin
    mov ecx,[B]
    mov [min],ecx
fin:
    mov eax,msg2
    call sprint
    mov eax,[min]
    call !printf
    call quit
1 Помощь 2 Сохран 3 Блок 4 Замена 5 Копия 6 Пере-ить 7 Поиск 8 Удалить 9 МенюМС 10 Выход

```

Рис. 3.17: Пишем программу

Транслируем файл и смотрим на работу программы (рис. fig. 3.18).

```

vlasovas@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
vlasovas@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
vlasovas@vbox:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 59
Наименьшее число: 59
vlasovas@vbox:~/work/arch-pc/lab07$

```

Рис. 3.18: Смотрим на работу программы(всё верно)

2. Напишите программу, которая для введенных с клавиатуры значений x и y вычисляет значение заданной функции $f(x, y)$ и выводит результат вычислений. Вид функции $f(x, y)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений

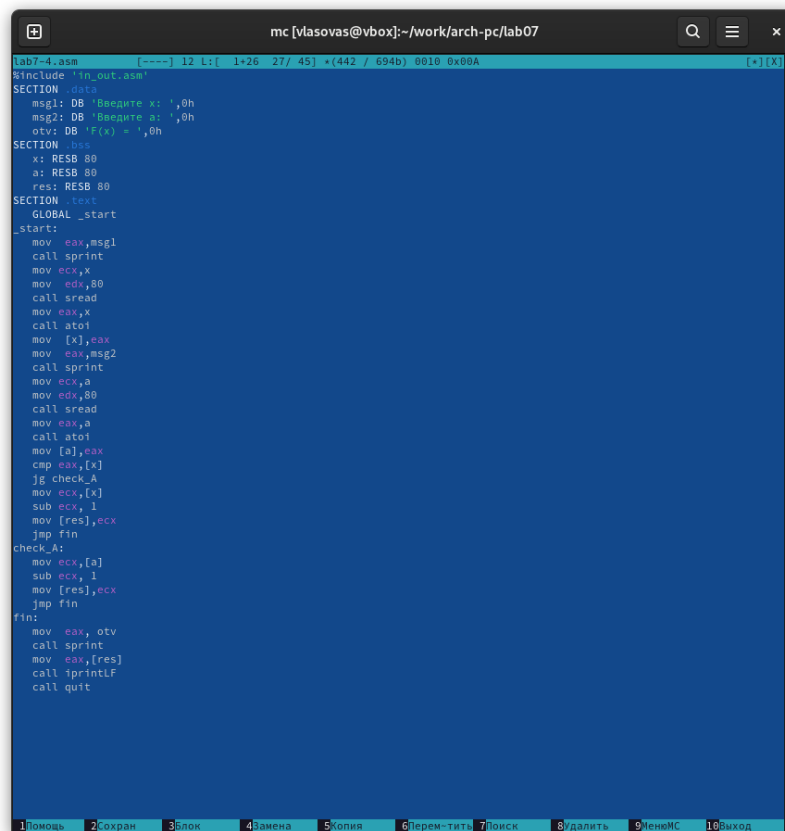
❌ и ❌ из 7.6.

Создаем новый файл (рис. fig. 3.19).

```
vlasovas@vbox:~/work/arch-pc/lab07$ touch lab7-4.asm
vlasovas@vbox:~/work/arch-pc/lab07$
```

Рис. 3.19: Создаем файл командой touch

Открываем его и пишем программу, которая решит систему уравнений, при данных, введенных в консоль (рис. fig. 3.20).



```
lab7-4.asm [-----] 12 L: [ 1+26 27/ 45] x(442 / 694b) 0010 0x00A [*)(X)
#include "in_out.asm"
SECTION .data
msg1: DB "Введите x: ",0h
msg2: DB "Введите a: ",0h
otv: DB "F(x) = ",0h
SECTION .bss
x: RESB 80
a: RESB 80
res: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,msg1
call sprint
mov ecx,x
mov edx,80
call sread
mov eax,x
call atoi
mov [x],eax
mov eax,msg2
call sprint
mov ecx,a
mov edx,80
call sread
mov eax,a
call atoi
mov [a],eax
cmp eax,[x]
jg check_A
mov ecx,[x]
sub ecx,1
mov [res],ecx
jmp fin
check_A:
mov ecx,[a]
sub ecx,1
mov [res],ecx
jmp fin
fin:
mov eax,otv
call sprint
mov eax,[res]
call iprintLF
call quit
```

Рис. 3.20: Пишем программу

Транслируем файл и проверяем его работу при $x=5$ и $a=7$ (рис. fig. 3.21).

```
vlasovas@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
vlasovas@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
vlasovas@vbox:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 5
Введите a: 7
F(x) = 6
```

Рис. 3.21: Проверяем работу программы

Транслируем файл и проверяем его работу при $x=6$ и $a=4$ (рис. fig. 3.22).

```
vlasovas@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
vlasovas@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
vlasovas@vbox:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 6
Введите a: 4
F(x) = 5
```

Рис. 3.22: Проверяем работу программы

4 Выводы

Мы познакомились с структурой файла листинга, изучили команды условного и безусловного перехода.