

Лабораторная работа №7

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

Баранов Никита Дмитриевич

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Самостоятельная работа	13
5	Выводы	16

Список иллюстраций

3.1	Создаем каталог и файл .asm	7
3.2	Вводим программу в файл	8
3.3	Создаем исполняемый файл и запускаем его	8
3.4	Изменяем текст программы	9
3.5	Создаем исполняемый файл и проверяем его работу	9
3.6	Редактируем текст программы для новых условий	10
3.7	Создаем исполняемый файл и проверяем его работу	10
3.8	Создаем файл lab7-2.asm	10
3.9	Записываем программу	11
3.10	Создаем исполняемый файл и проверяем работу	11
3.11	Создаем файл листинга и открываем в текстовом редакторе	12
3.12	Открываем файл и ознакомимся с форматом	12
4.1	Пишем программу для определения минимального значения	14
4.2	Создаем исполняемый файл и проверяем работу для значений 54,62,87	14
4.3	Пишем программу для нашей функции- номер 5	15
4.4	Создаем объектный файл и проверяем его работу на двух данных нам примерах	15

Список таблиц

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

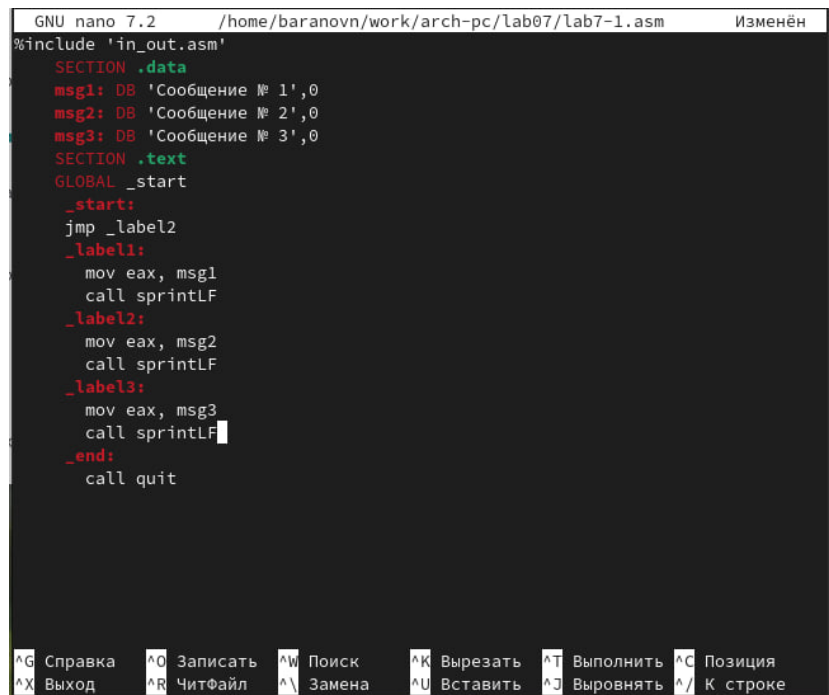
Написать программы для решения системы выражений.

3 Выполнение лабораторной работы

Создайте каталог для программ лабораторной работы № 7, перейдите в него и создайте файл lab7-1.asm. Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции jmp. Введите в файл lab7-1.asm текст программы из листинга 7.1. Создайте исполняемый файл и запустите его (рис. fig. 3.1) (рис. fig. 3.2) (рис. fig. 3.3).

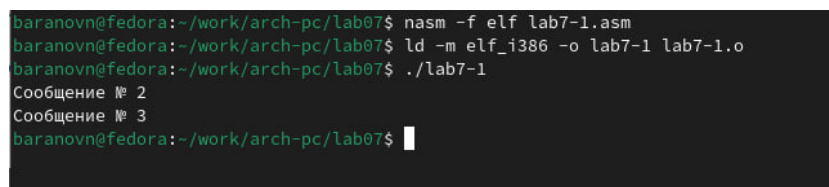
```
baranov@fedora:~$ mkdir ~/work/arch-pc/lab07
baranov@fedora:~$ cd ~/work/arch-pc/lab07
baranov@fedora:~/work/arch-pc/lab07$ touch lab7-1.asm
baranov@fedora:~/work/arch-pc/lab07$
```

Рис. 3.1: Создаем каталог и файл .asm



```
GNU nano 7.2 /home/baranovn/work/arch-pc/lab07/lab7-1.asm Изменён
%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
```

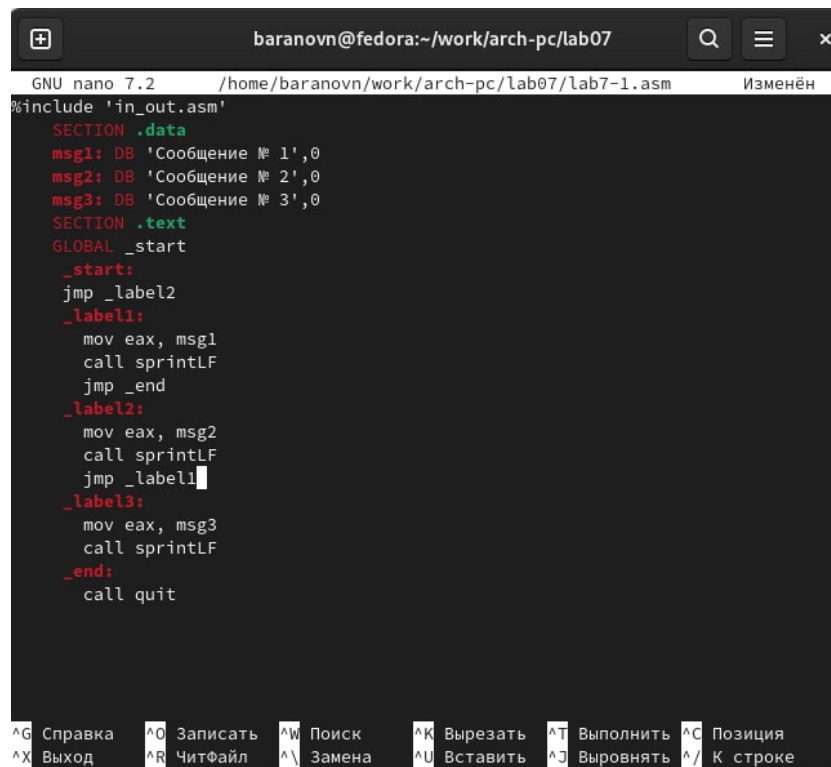
Рис. 3.2: Вводим программу в файл



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

Рис. 3.3: Создаем исполняемый файл и запускаем его

Наш вывод совпал с выводом в инструкции.Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу.Создайте исполняемый файл и проверьте его работу. Измените текст программы добавив или изменив инструкции jmp, чтобы вывод программы был Сообщение № 3 Сообщение № 2 Сообщение № 1.(рис. fig. 3.4)(рис. fig. 3.5)(рис. fig. 3.6)(рис. fig. 3.7)

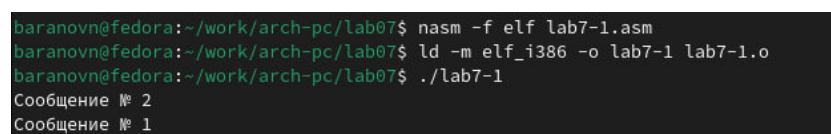


```
GNU nano 7.2 /home/baranovn/work/arch-pc/lab07/lab7-1.asm
%include 'in_out.asm'

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 sprintf
jmp _end
_label2:
mov eax, msg2
call sprintf
jmp _label1
_label3:
mov eax, msg3
call sprintf
_end:
call quit
```

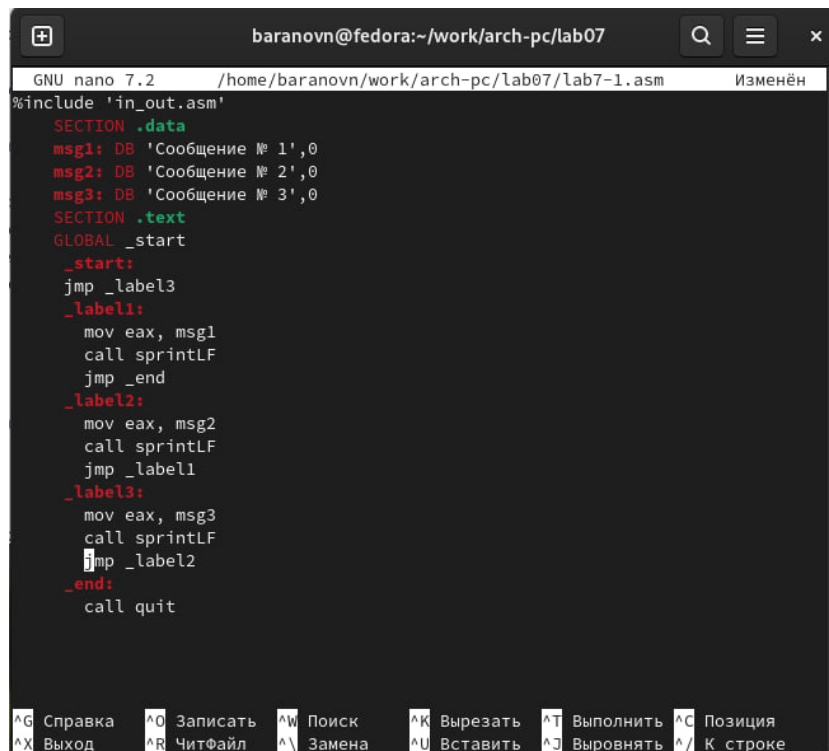
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^_ К строке

Рис. 3.4: Изменяем текст программы



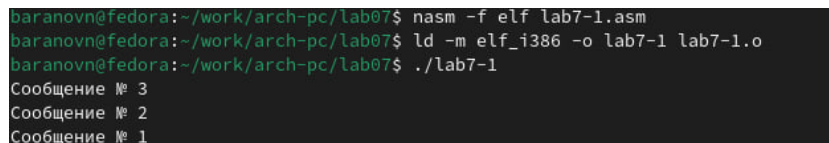
```
baranovn@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
baranovn@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
baranovn@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
```

Рис. 3.5: Создаем исполняемый файл и проверяем его работу



```
GNU nano 7.2 /home/baranovn/work/arch-pc/lab07/lab7-1.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 sprintLF
jmp _end
_label2:
mov eax, msg2
call sprintLF
jmp _label1
_label3:
mov eax, msg3
call sprintLF
jmp _label2
_end:
call quit
```

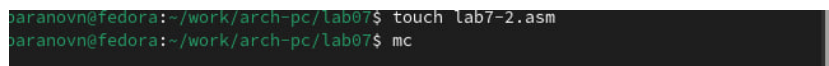
Рис. 3.6: Редактируем текст программы для новых условий



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

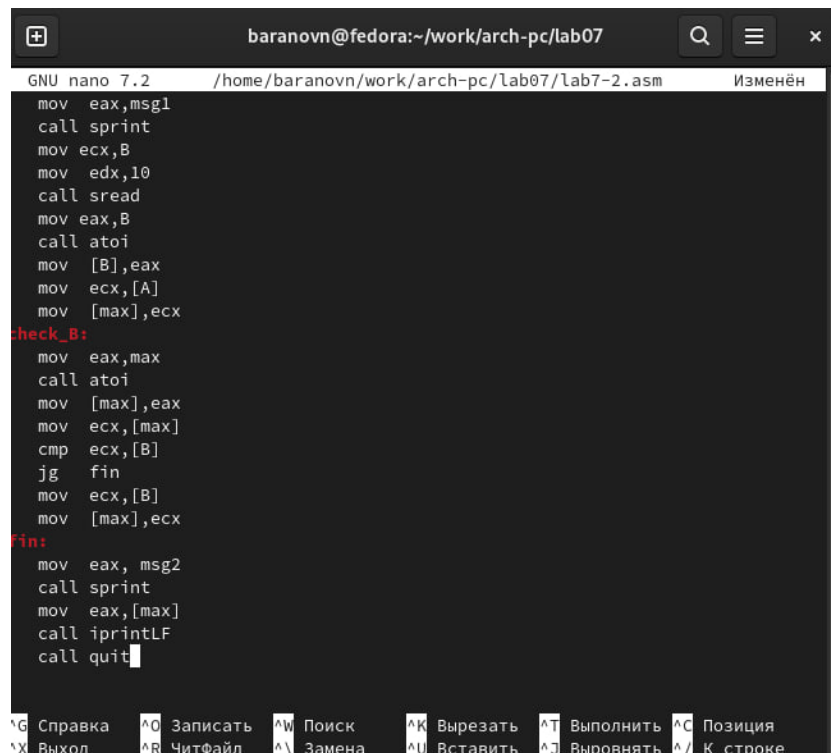
Рис. 3.7: Создаем исполняемый файл и проверяем его работу

Создайте файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Внимательно изучите текст программы из листинга 7.3 и введите в lab7-2.asm(рис. fig. 3.8)(рис. fig. 3.9)(рис. fig. 3.10)



```
baranovn@fedora:~/work/arch-pc/lab07$ touch lab7-2.asm
baranovn@fedora:~/work/arch-pc/lab07$ mc
```

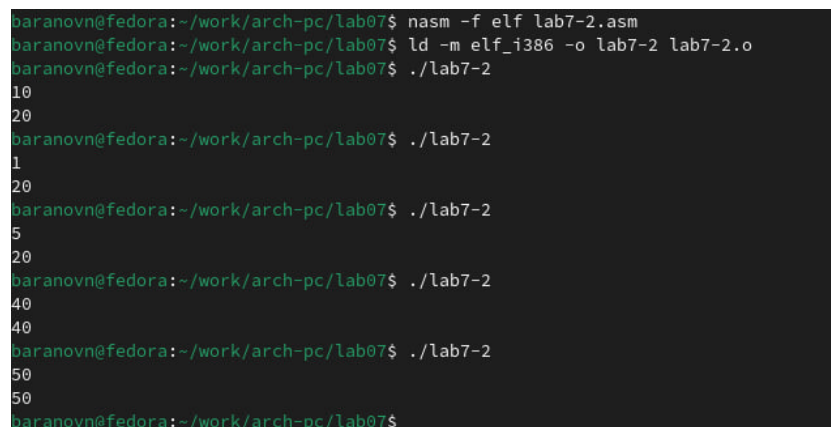
Рис. 3.8: Создаем файл lab7-2.asm



```
GNU nano 7.2 /home/baranovn/work/arch-pc/lab07/lab7-2.asm Изменён
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
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 Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выровнять ^/_ К строке

Рис. 3.9: Записываем программу



```
baranovn@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
baranovn@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
baranovn@fedora:~/work/arch-pc/lab07$ ./lab7-2
10
20
baranovn@fedora:~/work/arch-pc/lab07$ ./lab7-2
1
20
baranovn@fedora:~/work/arch-pc/lab07$ ./lab7-2
5
20
baranovn@fedora:~/work/arch-pc/lab07$ ./lab7-2
40
40
baranovn@fedora:~/work/arch-pc/lab07$ ./lab7-2
50
50
baranovn@fedora:~/work/arch-pc/lab07$
```

Рис. 3.10: Создаем исполняемый файл и проверяем работу

Создайте файл листинга для программы из файла lab7-2.asm. Откройте файл листинга lab7-2.lst с помощью любого текстового редактора. Внимательно ознакомьтесь с его форматом и содержимым. Подробно объяснить содержимое трёх строк файла листинга по выбору. Откройте файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалите один операнд. Выполните

трансляцию с получением файла листинга. Какие выходные файлы создаются в этом случае? Что добавляется в листинге? (рис. fig. 3.11) (рис. 3.12)

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

Рис. 3.11: Создаем файл листинга и открываем в текстовом редакторе

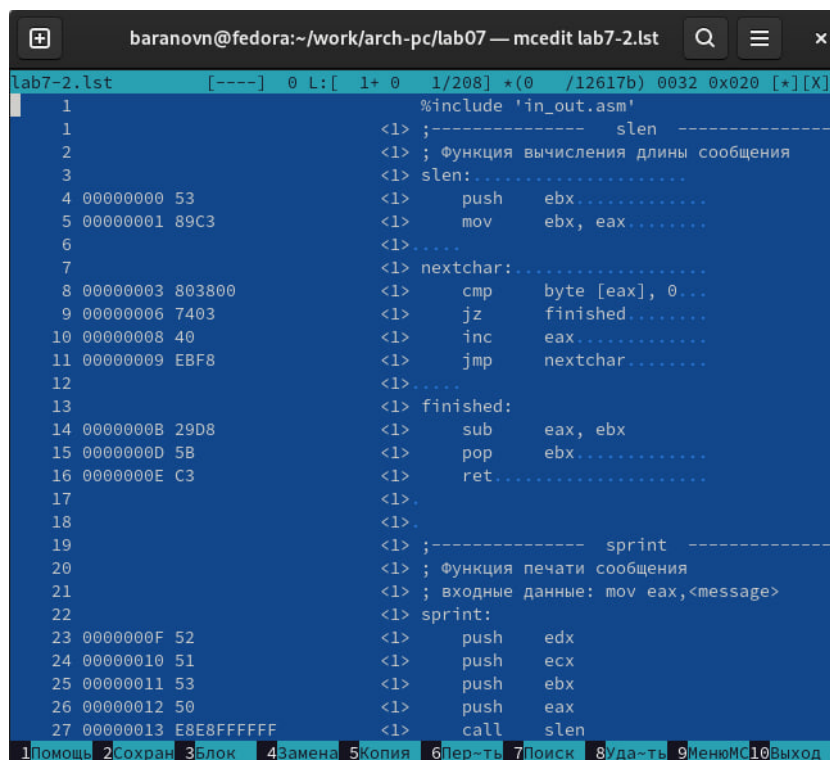
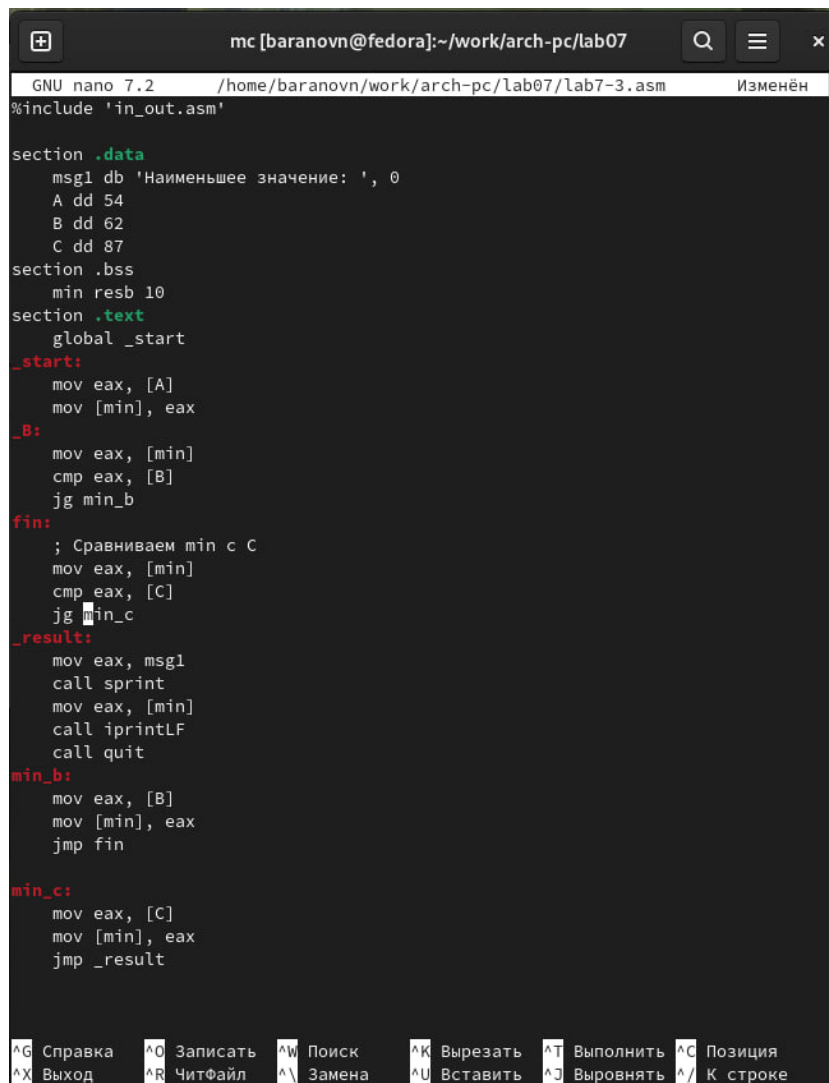


Рис. 3.12: Открываем файл и ознакомливается с форматом

Строка 5 - 00000001 89C3 mov ebx, eax 00000001 - адрес в сегменте кода 89C3 - машинный код для инструкции mov ebx, eax - присваивание переменной ebx значения, хранящегося в регистре eax
 Строка 26 - 00000012 50 push eax 00000012 - адрес в сегменте кода 50 - машинный код для инструкции push eax - значение из регистра eax помещается в стек
 Строка 53 - 0000003B E8CFFFFFFF call sprint 0000003B - адрес в сегменте кода E8CFFFFFFF - машинный код для инструкции call sprint - вызов функции sprint, которая выводит данные на экран

4 Самостоятельная работа

Напишите программу нахождения наименьшей из 3 целочисленных переменных a , b , c . Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу(рис. fig. 4.1)(рис. fig. 4.2)

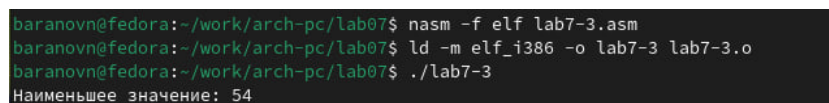


```
mc [baranovn@fedora]:~/work/arch-pc/lab07
GNU nano 7.2 /home/baranovn/work/arch-pc/lab07/lab7-3.asm
%include 'in_out.asm'

section .data
    msg1 db 'Наименьшее значение: ', 0
    A dd 54
    B dd 62
    C dd 87
section .bss
    min resb 10
section .text
    global _start
_start:
    mov eax, [A]
    mov [min], eax
_B:
    mov eax, [min]
    cmp eax, [B]
    jg min_b
fin:
    ; Сравниваем min с C
    mov eax, [min]
    cmp eax, [C]
    jg min_c
_result:
    mov eax, msg1
    call sprint
    mov eax, [min]
    call iprintLF
    call quit
min_b:
    mov eax, [B]
    mov [min], eax
    jmp fin
min_c:
    mov eax, [C]
    mov [min], eax
    jmp _result
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^_/ К строке

Рис. 4.1: Пишем программу для определения минимального значения



```
baranovn@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
baranovn@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
baranovn@fedora:~/work/arch-pc/lab07$ ./lab7-3
Наименьшее значение: 54
```

Рис. 4.2: Создаем исполняемый файл и проверяем работу для значений 54,62,87

Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ 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 x и a из 7.6.(рис. fig. 4.3)(рис. fig. 4.4)

```

GNU nano 7.2 lab7-4.asm
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 A

mov ecx, [x]
sub ecx, [a]
mov eax, ecx
add ecx, eax

xor edx, edx
mov eax, ecx
mov ebx, 10
div ebx
mov [res], eax
jmp fin

A:
mov ecx, 15
mov [res], ecx

fin:
mov eax, ans
call sprint
mov eax, [res]
call iprintLF
call quit

Сохранить изменённый буфер?
Y Да
N Нет
^C Отмена

```

Рис. 4.3: Пишем программу для нашей функции- номер 5

```

baranovn@fedora:~/work/arch-pc/lab07$ nano lab7-4.asm
baranovn@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
baranovn@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
baranovn@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 1
Введите a: 2
Результат: 15
baranovn@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 2
Введите a: 1
Результат: 2
baranovn@fedora:~/work/arch-pc/lab07$

```

Рис. 4.4: Создаем объектный файл и проверяем его работу на двух данных нам примерах

5 Выводы

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