

# **Отчет по лабораторной работе №7**

**Дисциплина: архитектура компьютера**

Гончарь Анастасия Александровна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
4.1	Реализация переходов в NASM . . . . .	8
4.2	Изучение структуры файлы листинга . . . . .	11
4.3	Задание для самостоятельной работы . . . . .	13
4.4	Листинг№1 для нахождения наименьшего из 3 чисел . . . . .	15
4.5	Листинг№2 для вычисления значения функции . . . . .	17
<b>5</b>	<b>Выводы</b>	<b>20</b>
	<b>Список литературы</b>	<b>21</b>

# Список иллюстраций

4.1	Создание каталога и файла lab8-1.asm . . . . .	8
4.2	Текст программы . . . . .	8
4.3	Запуск файла . . . . .	9
4.4	Изменение текста программы . . . . .	9
4.5	Запуск файла . . . . .	9
4.6	Изменение текста программы . . . . .	10
4.7	Запуск файла . . . . .	10
4.8	Создание файла . . . . .	10
4.9	Текст программы в файле . . . . .	11
4.10	Программа для сравнения чисел . . . . .	11
4.11	Файл листинга lab8-2.lst . . . . .	11
4.12	Объяснения третьей строки . . . . .	12
4.13	Создание файла без одного операнда . . . . .	12
4.14	Запуск файла . . . . .	12
4.15	Файл листинга без одного операнда . . . . .	13
4.16	Создание файла . . . . .	13
4.17	Текст программы в файле . . . . .	14
4.18	Результат работы программы . . . . .	14
4.19	Создание файла . . . . .	14
4.20	Текст программы в файле . . . . .	15
4.21	Результат работы программы . . . . .	15

# Список таблиц

3.1	Описание некоторых каталогов файловой системы GNU Linux . . .	7
-----	---	---

# 1 Цель работы

Целью данной лабораторной работы является изучение команд условного и безусловного переходов, приобретение навыков написания программ с использованием переходов и знакомство с назначением и структурой файла листинга.

## 2 Задание

1.Реализация переходов в NASM 2.Изучение структуры файлы листинга 3.Задание для самостоятельной работы

## 3 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы.

Например, в табл. 3.1 приведено краткое описание стандартных каталогов Unix.

Таблица 3.1: Описание некоторых каталогов файловой системы GNU Linux

Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую систему
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно про Unix см. в [1–4].

## 4 Выполнение лабораторной работы

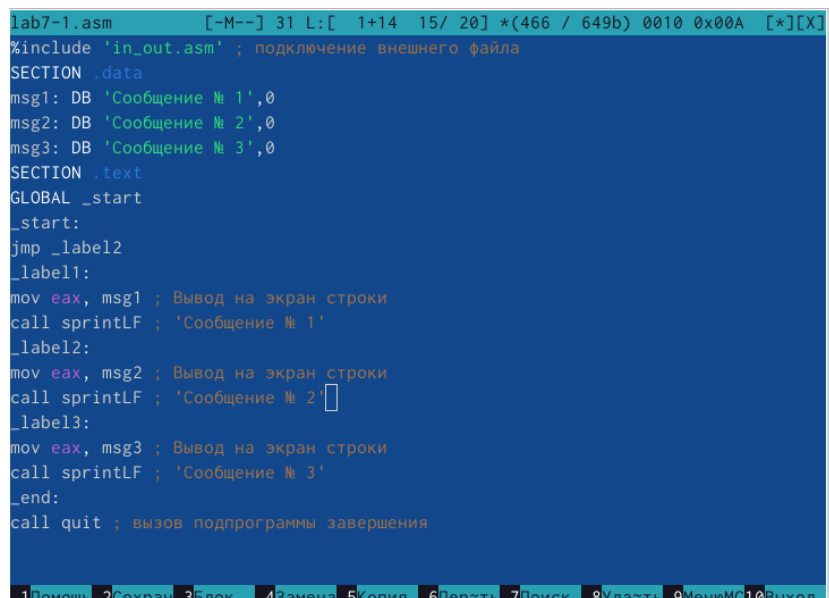
### 4.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы № 7, перехожу в него и создаю файл lab7-1.asm (рис. 4.1).

```
aagoncharj@dk8n76 ~ $ mkdir ~/work/arch-pc/lab07
aagoncharj@dk8n76 ~ $ cd ~/work/arch-pc/lab07
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ touch lab7-1.asm
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $
```

Рис. 4.1: Создание каталога и файла lab8-1.asm

Ввожу в файл lab7-1.asm текст программы из листинга 7.1 (рис. 4.2).



```
lab7-1.asm      [-M--] 31 L: [ 1+14 15/ 20] *(466 / 649b) 0010 0x00A [*][X]
%include 'in_out.asm' ; подключение внешнего файла
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 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.2: Текст программы

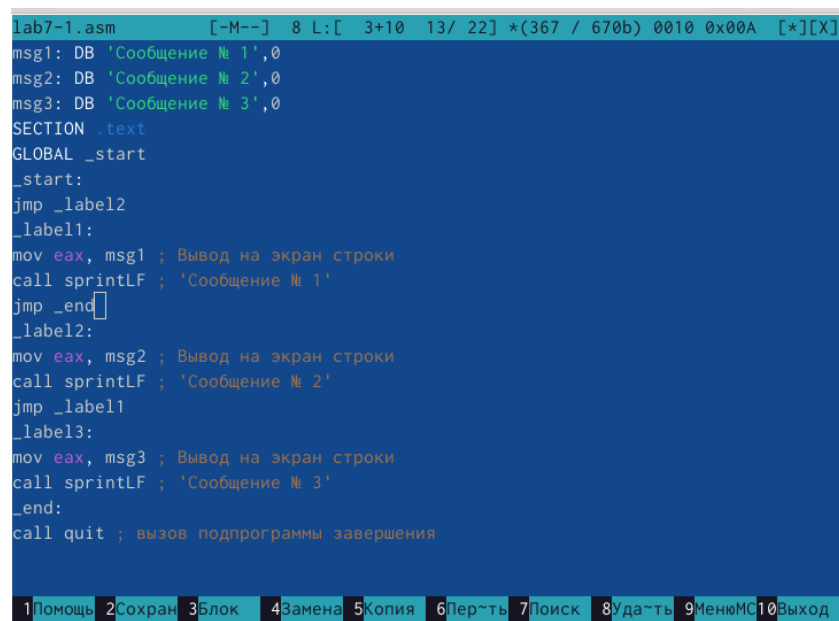


Создаю исполняемый файл и запускаю его (рис. 4.3).

```
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рис. 4.3: Запуск файла

Изменяю текст программы в файле lab7-1.asm в соответствии с листингом 7.2 (рис. 4.4)



```
lab7-1.asm [-M--] 8 L:[ 3+10 13/ 22] *(367 / 670b) 0010 0x00A [*][X]
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 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.4: Изменение текста программы

Создаю исполняемый файл и запускаю его (рис. 4.5).

```
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1
```

Рис. 4.5: Запуск файла

Теперь изменяю текст программы в этом же файле, чтобы программа выводила сначала “Сообщение №3”, затем “Сообщение №2” и “Сообщение №1” (рис. 4.6)

```

lab7-1.asm      [-M--] 11 L:[ 1+20 21/ 23] *(607 / 682b) 0010 0x00A  [*][X]
%include 'in_out.asm' ; подключение внешнего файла
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 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2
_end:

```

Рис. 4.6: Изменение текста программы

Создаю исполняемый файл и запускаю его (рис. 4.7).

```

aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ 

```

Рис. 4.7: Запуск файла

Создаю файл lab7-2.asm (рис. 4.8).

```

aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ touch lab7-2.asm
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ 

```

Рис. 4.8: Создание файла

Ввожу в этот файл текст программы из листинга 7.3 (рис. ??).

```

lab7-2.asm      [-M--]  9 L: [ 1+ 0  1/ 49] *(9  /1743b) 0039 0x027  [*][X]
#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

```

Рис. 4.9: Текст программы в файле

Создаю исполняемый файл и запускаю его (рис. 4.10).

```

aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm

aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 21
Наибольшее число: 50
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 44
Наибольшее число: 50
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 71
Наибольшее число: 71
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ 

```

Рис. 4.10: Программа для сравнения чисел

## 4.2 Изучение структуры файлы листинга

Создаю файл листинга для программы из файла lab7-2.asm с помощью ключа -l (рис. 4.11).

```

aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ 

```

Рис. 4.11: Файл листинга lab8-2.lst

Далее открываю файл lab7-2.lst для ознакомления с его форматом и содержанием (рис. 4.12).

```
lab7-2.lst      [----]  0 L: 1+ 0 1/226] *(0 /14544b) 0032 0x020 [*][X]
1               %include 'in_out.asm'
1               <1> ;----- slen -----
2               <1> ; Функция вычисления длины сообщения
3               <1> slen:.....
4 00000000 53    <1>      push    ebx.....
5 00000001 89C3  <1>      mov     ebx, eax.....
6               <1>.....
7               <1> nextchar:.....
8 00000003 803800 <1>      cmp     byte [eax], 0...
9 00000006 7403   <1>      jz      finished.....
10 00000008 40    <1>      inc     eax.....
11 00000009 EBF8  <1>      jmp     nextchar.....
12               <1>.....
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
```

Рис. 4.12: Объяснения третьей строки

Теперь в строке `mov eax, max` я убраю `max` в файле lab7-2.asm (рис. 4.13).

```
mov eax
```

Рис. 4.13: Создание файла без одного операнда

Пробую запустить файл, но программы выдает ошибку, так как для программы нужно два операнда (рис. 4.14).

```
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:14: error: invalid combination of opcode and operands
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $
```

Рис. 4.14: Запуск файла

В файле листинга тоже отображается ошибка (рис. 4.15).

```

lab7-2.lst      [----]  0 L:[182+ 8 190/226] *(11592/14544b) 0032 0x020[*][X]
7              section .bss
8 00000000 <res Ah>      max resb 10
9 0000000A <res Ah>      B resb 10
10             section .text
11             global _start
12             _start:
13             ; ----- Вывод сообщения 'Введите B:
14             mov eax
14             *****      error: invalid combination of opcode and
15 000000E8 E822FFFFFF      call sprint
16             ; ----- Ввод 'B'
17 000000ED B9[0A000000]    mov ecx,B
18 000000F2 BA0A000000      mov edx,10
19 000000F7 E847FFFFFF      call sread
20             ; ----- Преобразование 'B' из символа
21 000000FC B8[0A000000]    mov eax,B
22 00000101 E896FFFFFF      call atoi ; Вызов подпрограммы перевода
23 00000106 A3[0A000000]    mov [B],eax ; запись преобразованного числа
24             ; ----- Записываем 'A' в переменную
25 0000010B 8B0D[35000000]  mov ecx,[A] ; 'ecx = A'
26 00000111 890D[00000000]  mov [max],ecx ; 'max = A'
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход

```

Рис. 4.15: Файл листинга без одного операнда

## 4.3 Задание для самостоятельной работы

1.Сначала я создаю файл lab7-3.asm (рис. 4.16).

```

aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ touch lab7-3.asm
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ mc

```

Рис. 4.16: Создание файла

Так как я меня 15 вариант, я написала программу для нахождения наименьшего из трех чисел для 32, 6, 54 (рис. 4.17).

```

lab7-3.asm  [-M--] 26 L: [ 1+13 14/ 41] *(254 /1515b) 1072 0x430  [*][X]
#include "in_out.asm"
section .data
msg1 db "Введите B: ",0h
msg2 db "Наименьшее число: ",0h
A dd '32'
B dd '6'
C dd '54'
section .bss
min resb 10
section .text
global _start
_start:
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jl check_B ; если 'A<C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'
; ----- Преобразование 'min(A,C)' из символа в число
check_B:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в 'min'
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
jl fin ; если 'min(A,C)<B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наименьшее число: '
mov eax,[min]
call iprintf ; Вывод 'min(A,B,C)'
call quit ; Выход

```

Рис. 4.17: Текст программы в файле

Теперь создаю исполняемый файл и запускаю его. Программа вывела меньшее из этих чисел (рис. 4.18).

```

aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $ ./lab7-3
Наименьшее число: 6
aagoncharj@dk8n76 ~/work/arch-pc/lab07 $

```

Рис. 4.18: Результат работы программы

2.Сначала я создаю файл lab7-4.asm (рис. 4.19).

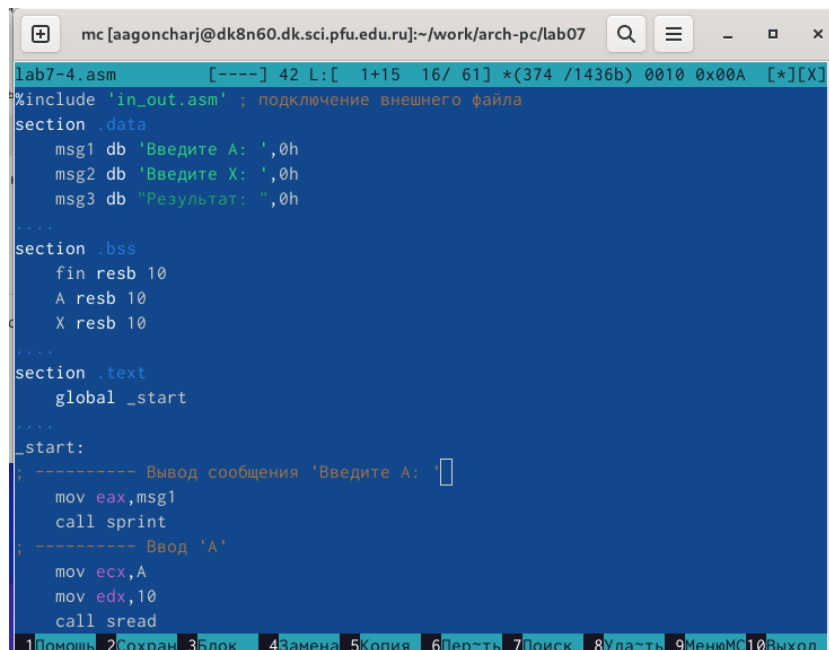
```

aagoncharj@dk4n65 ~/work/arch-pc/lab07 $ touch lab7-4.asm

```

Рис. 4.19: Создание файла

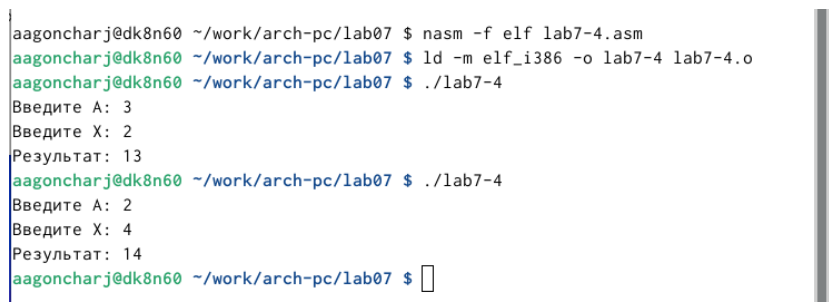
Теперь мне необходимо написать программу, для вычисления значения функции при введенных X и A (рис. 4.20).



```
lab7-4.asm      [----] 42 L: [ 1+15 16/ 61] *(374 /1436b) 0010 0x00A [*][X]
%include 'in_out.asm' ; подключение внешнего файла
section .data
    msg1 db 'Введите A: ',0h
    msg2 db 'Введите X: ',0h
    msg3 db 'Результат: ',0h
    ....
section .bss
    fin resb 10
    A resb 10
    X resb 10
    ....
section .text
    global _start
    ....
_start:
; ----- Вывод сообщения 'Введите A: '
    mov eax,msg1
    call sprint
; ----- Ввод 'A'
    mov ecx,A
    mov edx,10
    call sread
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход
```

Рис. 4.20: Текст программы в файле

Теперь создаю исполняемый файл и запускаю его. Программа работает верно (рис. 4.21).



```
aagoncharj@dk8n60 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm
aagoncharj@dk8n60 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
aagoncharj@dk8n60 ~/work/arch-pc/lab07 $ ./lab7-4
Введите A: 3
Введите X: 2
Результат: 13
aagoncharj@dk8n60 ~/work/arch-pc/lab07 $ ./lab7-4
Введите A: 2
Введите X: 4
Результат: 14
aagoncharj@dk8n60 ~/work/arch-pc/lab07 $
```

Рис. 4.21: Результат работы программы

## 4.4 Листинг№1 для нахождения наименьшего из 3 чисел

```
%include 'in_out.asm'
section .data
```

```

msg1 db 'Введите B: ',0h
msg2 db "Наименьшее число: ",0h
A dd '32'
B dd '6'
C dd '54'

section .bss
min resb 10

section .text
global _start
_start:
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jl check_B ; если 'A<C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'
; ----- Преобразование 'min(A,C)' из символа в число
check_B:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в 'min'
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'

```



```

jl fin ; если 'min(A,C)<B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наименьшее число: '
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход

```

## 4.5 Листинг№2 для вычисления значения функции

```

#include 'in_out.asm' ; подключение внешнего файла
section .data
    msg1 db 'Введите A: ',0h
    msg2 db 'Введите X: ',0h
    msg3 db "Результат: ",0h

section .bss
    fin resb 10
    A resb 10
    X resb 10

section .text
    global _start

_start:
; ----- Вывод сообщения 'Введите A: '

```

```

    mov eax,msg1
    call sprint
; ----- Ввод 'A'
    mov ecx,A
    mov edx,10
    call sread
; ----- Преобразование 'A' из символа в число
    mov eax,A
    call atoi ; Вызов подпрограммы перевода символа в число
    mov [A],eax

; ----- Вывод сообщения 'Введите X: '
    mov eax,msg2
    call sprint
; ----- Ввод 'X'
    mov ecx,X
    mov edx,10
    call sread
; ----- Преобразование 'X' из символа в число
    mov eax,X
    call atoi ; Вызов подпрограммы перевода символа в число
    mov [X],eax

    mov ecx,[X]
    mov ebx,[A]
    cmp ecx,ebx
    jge func2
    mov edx,[A]

```

```
add edx,10
mov [fin],edx
jmp final
```

func2:

```
mov ax,[X]
add ax,10
mov [fin],ax
jmp final
```

final:

```
mov eax,msg3
call sprint ; Вывод сообщения 'Результат: '
mov eax,[fin]
call iprintLF ; Вывод fin
call quit ; Выход
```

## **5 Выводы**

При выполнении данной лабораторной работы я изучила команды условного и безусловного переходов, приобрела навыки написания программ с использованием переходов и ознакомилась с назначением и структурой файла листинга.

## Список литературы

1. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.
2. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.