

# **Лабораторная работа №8.**

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

Боровиков Даниил Александрович

# Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Выводы	24

## Список иллюстраций

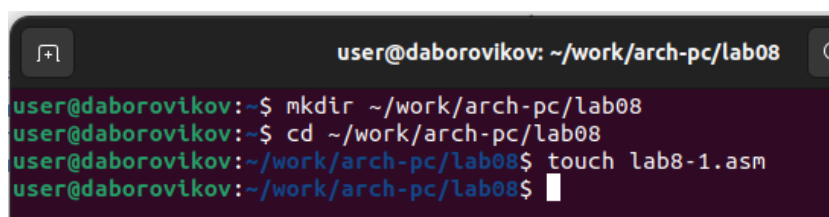
2.1	Создание файла lab8-1.asm в соответствующем каталоге . . . . .	5
2.2	Текст программы из листинга 8.1. . . . .	5
2.3	Запуск исполняемого файла lab8-1.asm . . . . .	6
2.4	Текст измененной программы . . . . .	6
2.5	Запуск исправленного исполняемого файла lab8-1.asm . . . . .	6
2.6	Листинг программы с требуемым выводом . . . . .	7
2.7	Запуск измененного исполняемого файла lab8-1.asm . . . . .	7
2.8	Листинга 8.3 . . . . .	8
2.9	Запуск исполняемого файла lab8-2.asm . . . . .	9
2.10	Создание файла листинга . . . . .	9
2.11	Листинг программы . . . . .	10
2.12	Ошибка трансляции в термине . . . . .	11
2.13	Вывод ошибки в листинге . . . . .	11
2.14	Текст программы . . . . .	12
2.15	Текст программы . . . . .	13
2.16	Проверка работы исполняемого файла . . . . .	13
2.17	Листинг программы lab8-4.asm . . . . .	18
2.18	Проверка работы программы . . . . .	19

# 1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

## 2 Выполнение лабораторной работы

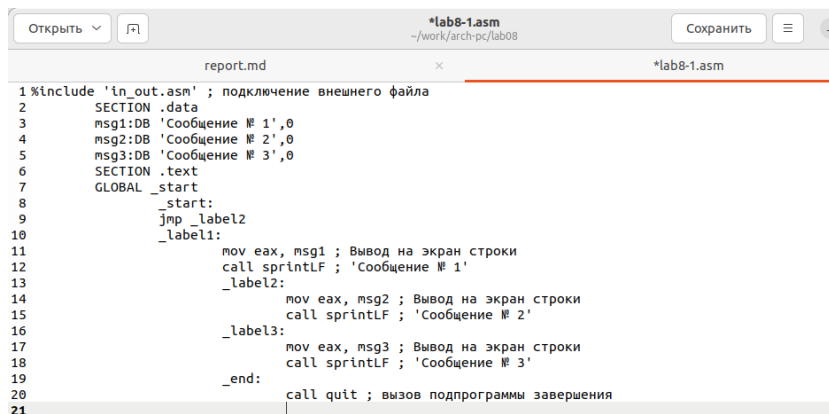
Создадим каталог для программ лабораторной работы № 8, перейдем в него и создадим файл lab8-1.asm(рис. 2.1)



```
user@daborovikov: ~/work/arch-pc/lab08
user@daborovikov:~$ mkdir ~/work/arch-pc/lab08
user@daborovikov:~$ cd ~/work/arch-pc/lab08
user@daborovikov:~/work/arch-pc/lab08$ touch lab8-1.asm
user@daborovikov:~/work/arch-pc/lab08$
```

Рис. 2.1: Создание файла lab8-1.asm в соответствующем каталоге

Введем в файл lab8-1.asm текст программы из листинга 8.1.(рис. 2.2)



```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3     msg1:DB 'Сообщение № 1',0
4     msg2:DB 'Сообщение № 2',0
5     msg3:DB 'Сообщение № 3',0
6 SECTION .text
7     GLOBAL _start
8     _start:
9         jmp _label2
10    _label1:
11        mov eax, msg1 ; Вывод на экран строки
12        call sprintf ; 'Сообщение № 1'
13    _label2:
14        mov eax, msg2 ; Вывод на экран строки
15        call sprintf ; 'Сообщение № 2'
16    _label3:
17        mov eax, msg3 ; Вывод на экран строки
18        call sprintf ; 'Сообщение № 3'
19    _end:
20        call quit ; вызов подпрограммы завершения
21
```

Рис. 2.2: Текст программы из листинга 8.1.

Создадим исполняемый файл и запустим его.(рис. 2.3)

```

user@daborovikov:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
user@daborovikov:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
user@daborovikov:~/work/arch-pc/lab08$ ./lab8-1
Сообщение № 2
Сообщение № 3
user@daborovikov:~/work/arch-pc/lab08$

```

Рис. 2.3: Запуск исполняемого файла lab8-1.asm

Далее изменим текст программы в соответствии с листингом 8.2(рис. 2.4)

```

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3     msg1:DB 'Сообщение № 1',0
4     msg2:DB 'Сообщение № 2',0
5     msg3:DB 'Сообщение № 3',0
6 SECTION .text
7     GLOBAL _start
8     _start:
9         jmp _label2
10    _label1:
11        mov eax, msg1 ; Вывод на экран строки
12        call sprintf ; 'Сообщение № 1'
13        jmp _end
14    _label2:
15        mov eax, msg2 ; Вывод на экран строки
16        call sprintf ; 'Сообщение № 2'
17        jmp _label1
18    _label3:
19        mov eax, msg3 ; Вывод на экран строки
20        call sprintf ; 'Сообщение № 3'
21    _end:
22        call quit ; вызов подпрограммы завершения
23

```

Рис. 2.4: Текст измененной программы

Создадим исполняемый файл исправленного текста программы lab8-1.asm и запустите его.(рис. 2.5)

```

user@daborovikov:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
user@daborovikov:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
user@daborovikov:~/work/arch-pc/lab08$ ./lab8-1
Сообщение № 2
Сообщение № 1

```

Рис. 2.5: Запуск исправленного исполняемого файла lab8-1.asm

Текст программы для следующего вывода:

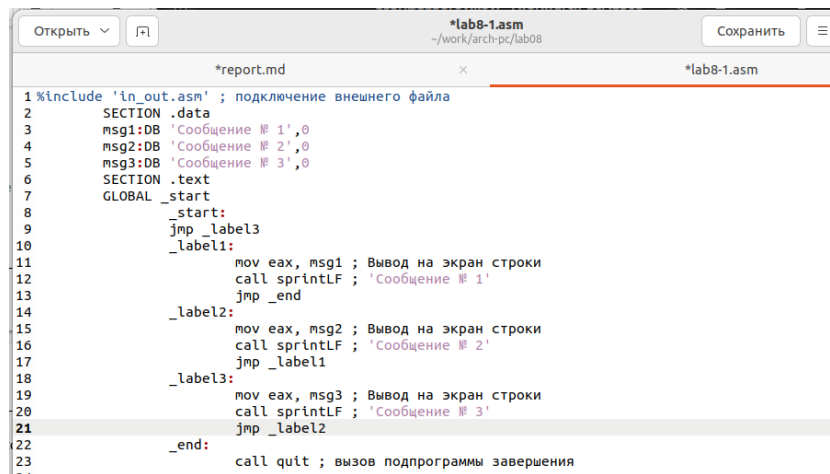
user@dk4n31:~\$ ./lab8-1

Сообщение № 3

Сообщение № 2

Сообщение № 1

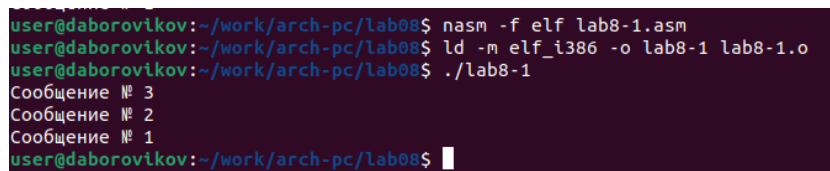
user@dk4n31:~\$(рис. 2.6)



```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1:DB 'Сообщение № 1',0
4 msg2:DB 'Сообщение № 2',0
5 msg3:DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label3
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintf ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintf ; 'Сообщение № 3'
21 jmp _label2
22 _end:
23 call quit ; вызов подпрограммы завершения
```

Рис. 2.6: Листинг программы с требуемым выводом

Создадим исполняемый файл и запустим его(рис. 2.7)



```
user@daborovikov:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
user@daborovikov:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
user@daborovikov:~/work/arch-pc/lab08$ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
user@daborovikov:~/work/arch-pc/lab08$
```

Рис. 2.7: Запуск измененного исполняемого файла lab8-1.asm

Создадим файл lab8-2.asm в каталоге ~/work/arch-pc/lab08. Внимательно изучим текст программы из листинга 8.3 и введем в lab8-2.asm.(рис. 2.8)

```

1 %include 'in_out.asm'
2 section .data
3     msg1 db 'Введите B: ',0h
4     msg2 db "Наибольшее число: ",0h
5     A dd '20'
6     C dd '50'
7 section .bss
8     max resb 10
9     B resb 10
10 section .text
11     global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14     mov eax,msg1
15     call sprint
16 ; ----- Ввод 'B'
17
18     mov ecx,B
19     mov edx,10
20     call sread
21 ; ----- Преобразование 'B' из символа в число
22     mov eax,B
23     call atoi ; Вызов подпрограммы перевода символа в число
24     mov [B],eax ; запись преобразованного числа в 'B'
25 ; ----- Записываем 'A' в переменную 'max'
26     mov ecx,[A] ; 'ecx = A'
27     mov [max],ecx ; 'max = A'
28 ; ----- Сравниваем 'A' и 'C' (как символы)
29     cmp ecx,[C] ; Сравниваем 'A' и 'C'
30     jg check_B; если 'A>C', то переход на метку 'check_B',
31     mov ecx,[C] ; иначе 'ecx = C'
32     mov [max],ecx ; 'max = C'
33 ; ----- Преобразование 'max(A,C)' из символа в число
34
35 check_B:
36     mov eax,max
37     call atoi ; Вызов подпрограммы перевода символа в число
38     mov [max],eax ; запись преобразованного числа в 'max'
39 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
40     mov ecx,[max] ; cmp ecx,[B]; Сравниваем 'max(A,C)' и 'B'
41     jg fin; если 'max(A,C)>B', то переход на 'fin',
42     mov ecx,[B] ; иначе 'ecx = B'
43     mov [max],ecx
44     ; ----- Вывод результата
45 fin:
46     mov eax, msg2
47     call sprint ; Вывод сообщения 'Наибольшее число: '
48     mov eax,[max]
49     call iprintfLF; Вывод 'max(A,B,C)'
50     call quit; Выход

```

Рис. 2.8: Листинг 8.3

Создадим измененный исполняемый файл и запустим его(рис. 2.9)



```

user@daborovikov:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
user@daborovikov:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
user@daborovikov:~/work/arch-pc/lab08$ ./lab8-2
Введите В: 12
Наибольшее число: 50
user@daborovikov:~/work/arch-pc/lab08$ ./lab8-2
Введите В: 100
Наибольшее число: 100
user@daborovikov:~/work/arch-pc/lab08$ ./lab8-2
Введите В: 32
Наибольшее число: 50
user@daborovikov:~/work/arch-pc/lab08$

```

Рис. 2.9: Запуск исполняемого файла lab8-2.asm

Создадим файл листинга для программы из файла lab8-2.asm и откроем файл при помощи текстового редактора mcedit(рис. 2.10)

```

user@daborovikov:~/work/arch-pc/lab08$ nasm -f elf -l lab8-2.lst lab8-2.asm
user@daborovikov:~/work/arch-pc/lab08$ mcedit lab8-2.lst
user@daborovikov:~/work/arch-pc/lab08$

```

Рис. 2.10: Создание файла листинга

Разберем три строки листинга программы

21 00000101 B8[0A000000] mov eax,B

Значение строки:

21-номер строки в коде листинга от начала сегмента

00000101 - адрес

B8[0A000000] - машинный код(B8[0A000000] - инструкция mov eax,B ; B8 - обозна

mov eax,B - исходный текст программы

22 00000106 E891FFFFFF call atoi ; Вызов подпрограммы перевода символа в число

Значение строки:

22-номер строки в коде листинга от начала сегмента

00000106 - адрес

E891FFFFFF - машинный код(E891FFFFFF - инструкция call atoi ; E8 - обозначает

call atoi ; Вызов подпрограммы перевода символа в число - исходный текст прог

23 0000010B A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'

Значение строки:

23-номер строки в коде листинга от начала сегмента

0000010B - адрес

A3[0A000000] - машинный код(A3[0A000000] - инструкция mov [B],eax ; A3 - обоз

mov eax,B - исходный текст программы

(рис. 2.11)

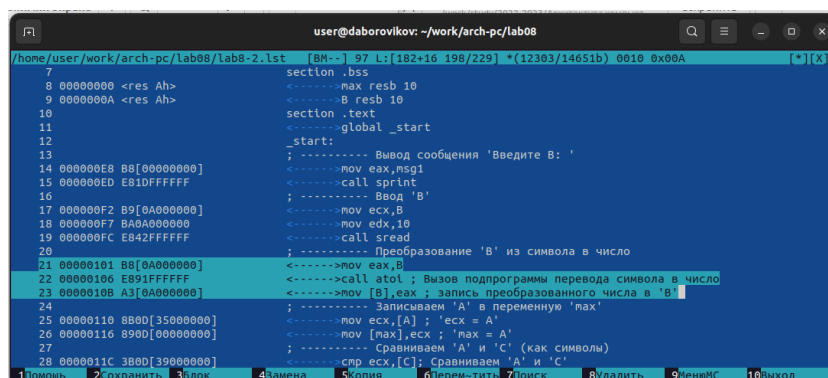


Рис. 2.11: Листинг программы

Откроем файл с программой lab8-2.asm и в любой инструкции с двумя операндами удалим один операнд. Выполните трансляцию с получением файла листинга:(рис. 2.12)

```

user@daborovikov:~/work/arch-pc/lab08$ nasm -f elf -l lab8-2.lst lab8-2.asm
lab8-2.asm:46: error: invalid combination of opcode and operands
user@daborovikov:~/work/arch-pc/lab08$

```

Рис. 2.12: Ошибка трансляции в терминале

(рис. 2.13)

```

221    46                                mov eax
222    46    ***** error: invalid combination of opcode

```

Рис. 2.13: Вывод ошибки в листинге

На выходе получаем листинг lab8-2.lst с ошибкой 46 \*\*\*\*\* error: invalid combination of opcode and operands

#Самостоятельная работа

Мой вариант номер 7

Напишем программу нахождения наименьшей из 3 целочисленных переменных А, В, С. Значения переменных выберем из табл. 8.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создадим исполняемый файл и проверьте его работу.(рис. 2.14)

```
1 %include 'in_out.asm'
2 section .data
3     msg1 db 'Введите B: ',0h
4     msg2 db "Наименьшее число: ",0h
5     msg3 db 'Введите A: ',0h
6     msg4 db 'Введите C: ',0h
7 section .bss
8     min resb 10
9     B resb 10
10    A resb 10
11    C resb 10
12 section .text
13     global _start
14 _start:
15 ; ----- Вывод сообщения 'Введите A: '
16     mov eax,msg3
17     call sprint
18 ; ----- Ввод 'A'
19     mov ecx,A
20     mov edx,10
21     call sread
22 ; ----- Преобразование 'a' из символа в число
23     mov eax,A
24     call atoi ; Вызов подпрограммы перевода символа в число
25     mov [A],eax ; запись преобразованного числа в 'B'
26 ; ----- Вывод сообщения 'Введите B: '
27     mov eax,msg1
28     call sprint
29 ; ----- Ввод 'B'
30     mov ecx,B
31     mov edx,10
32     call sread
33 ; ----- Преобразование 'b' из символа в число
34     mov eax,B
35     call atoi ; Вызов подпрограммы перевода символа в число
36     mov [B],eax ; запись преобразованного числа в 'B'
37 ; ----- Вывод сообщения 'Введите C: '
38     mov eax,msg4
39     call sprint
40 ; ----- Ввод 'C'
41     mov ecx,C
42
```

Рис. 2.14: Текст программы

Текст программы (рис. 2.15)

```

33
34 ; ----- Преобразование 'b' из символа в число
35 mov eax,B
36 call atoi ; Вызов подпрограммы перевода символа в число
37 mov [B],eax ; запись преобразованного числа в 'B'
38 ; ----- Вывод сообщения 'Введите C: '
39 mov eax,msg4
40 call sprint
41 ; ----- Ввод 'C'
42 mov ecx,C
43 mov edx,10
44 call sread
45 ; ----- Преобразование 'C' из символа в число
46 mov eax,C
47 call atoi ; Вызов подпрограммы перевода символа в число
48 mov [C],eax ; запись преобразованного числа в 'B'
49 ; ----- Записываем 'A' в переменную 'min'
50 mov ecx,[A] ; 'ecx = A'
51 mov [min],ecx ; 'min = A'
52 ; ----- Сравниваем 'A' и 'C' (как символы)
53 cmp [C],ecx ; Сравниваем 'C' и 'A'
54 jg check_B ; если 'C>A', то переход на метку 'check_B', то есть сравним с B
55 mov ecx,[C] ; иначе 'ecx = C'
56 mov [min],ecx ; 'min = C'
57 ; -----Преобразование 'min(A,C)' из символа в число
58 check_B:
59 ; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
60 mov ecx,[min]
61 cmp [B],ecx ; Сравниваем 'B' and 'min(A,C)'
62 jg fin ; если 'min(A,C)<B', то переход на 'fin',
63 mov ecx,[B] ; иначе 'ecx = B'
64 mov [min],ecx
65 ; -----Вывод результата
66 fin:
67 mov eax, msg2
68 call sprint ; Вывод сообщения 'Наибольшее число: '
69 mov eax,[min]
70 call iprintf; Вывод 'min(A,B,C)'
71 call quit; Выход
72
73
74

```

Рис. 2.15: Текст программы

Проверка работы исполняемого файла(рис. 2.16)

```

user@daborovikov:~/work/arch-pc/lab08$ ./lab8-3
Введите A: 45
Введите B: 67
Введите C: 15
Наименьшее число: 15
user@daborovikov:~/work/arch-pc/lab08$

```

Рис. 2.16: Проверка работы исполняемого файла

Листинг программы:

%include 'in\_out.asm'

section .data

msg1 db 'Введите B: ',0h

```

msg2 db "Наименьшее число: ",0h

msg3 db 'Введите A: ',0h

msg4 db 'Введите C: ',0h

section .bss

min resb 10

B resb 10

A resb 10

C resb 10

section .text

global _start

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

mov eax,msg3

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 ; запись преобразованного числа в 'B'
```

```
; ----- 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'
```

```

;----- Вывод сообщения 'Введите С:'

mov eax,msg4

call sprint

;----- Ввод 'С'

mov ecx,C

mov edx,10

call sread

; ----- Преобразование 'С' из символа в число

mov eax,C

call atoi ; Вызов подпрограммы перевода символа в число

mov [C],eax ; запись преобразованного числа в 'В'

;----- Записываем 'А' в переменную 'min'

mov ecx,[A] ; 'ecx = A'

mov [min],ecx ; 'min = A'

;----- Сравниваем 'А' и 'С' (как символы)

cmp [C],ecx; Сравниваем 'С' и 'А'

jg check_B; если 'C>A', то переход на метку 'check_B', то есть сравним с б

```



```

mov ecx,[C]; иначе 'ecx = C'

mov [min],ecx ; 'min = C'

;-----Преобразование 'min(A,C)' из символа в число
check_B:
;----- Сравниваем 'min(A,C)' и 'B' (как числа)

mov ecx,[min]

cmp [B],ecx; Сравниваем 'B' and 'min(A,C)'

jg fin

mov ecx,[B] ; иначе 'ecx = B'

mov [min],ecx

; -----Вывод результата

fin:

mov eax, msg2

call sprint ; Вывод сообщения 'Наибольшее число: '

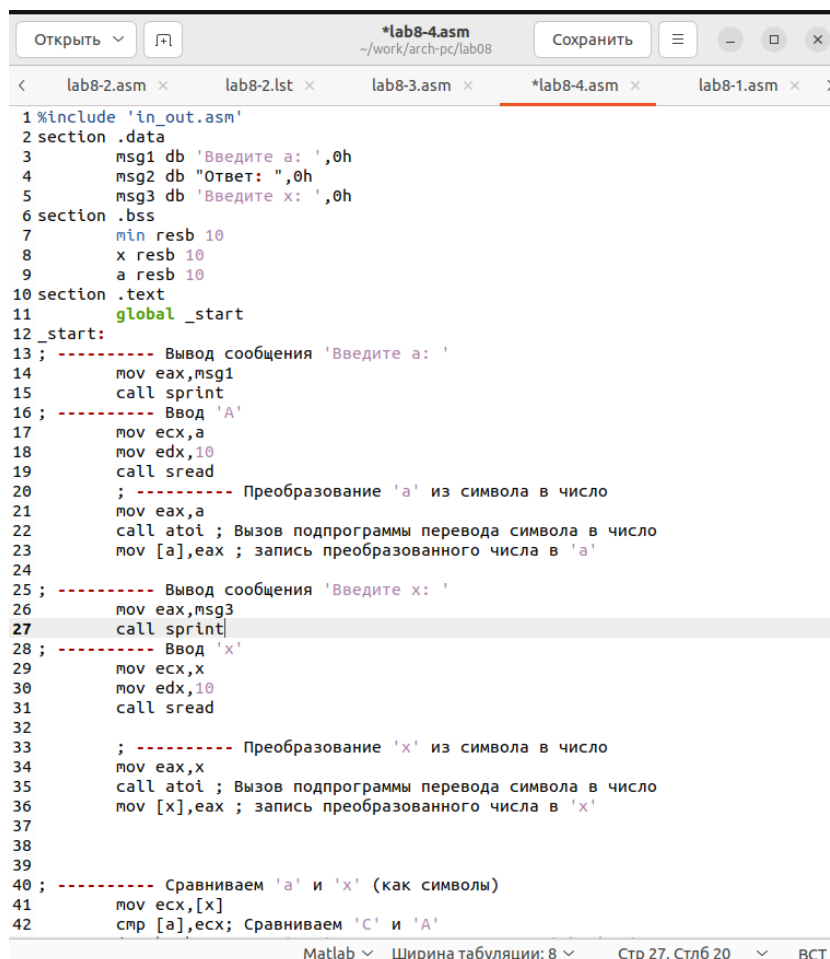
mov eax,[min]

call iprintLF; Вывод 'min(A,B,C)'

call quit; Выход

```

Напишем программу, которая для введенных с клавиатуры значений  $\boxed{x}$  и  $\boxed{a}$  вычисляет значение заданной функции  $\boxed{f(x)}$  и выводит результат вычислений. Вид функции  $\boxed{f(x)}$  выбрать из таблицы 8.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений  $\boxed{x}$  и  $\boxed{a}$  из 8.6.(рис. 2.17)



```

1 %include 'in_out.asm'
2 section .data
3     msg1 db 'Введите a: ',0h
4     msg2 db "Ответ: ",0h
5     msg3 db 'Введите x: ',0h
6 section .bss
7     min resb 10
8     x resb 10
9     a resb 10
10 section .text
11     global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите a: '
14     mov eax,msg1
15     call sprint
16 ; ----- Ввод 'a'
17     mov ecx,a
18     mov edx,10
19     call sread
20 ; ----- Преобразование 'a' из символа в число
21     mov eax,a
22     call atoi ; Вызов подпрограммы перевода символа в число
23     mov [a],eax ; запись преобразованного числа в 'a'
24
25 ; ----- Вывод сообщения 'Введите x: '
26     mov eax,msg3
27     call sprint
28 ; ----- Ввод 'x'
29     mov ecx,x
30     mov edx,10
31     call sread
32
33 ; ----- Преобразование 'x' из символа в число
34     mov eax,x
35     call atoi ; Вызов подпрограммы перевода символа в число
36     mov [x],eax ; запись преобразованного числа в 'x'
37
38
39
40 ; ----- Сравниваем 'a' и 'x' (как символы)
41     mov ecx,[x]
42     cmp [a],ecx; Сравниваем 'c' и 'A'

```

Рис. 2.17: Листинг программы lab8-4.asm

Листинг программы lab8-4.asm(рис. ??)

```

27      call sprint
28 ; ----- Ввод 'x'
29      mov ecx,x
30      mov edx,10
31      call sread
32
33      ; ----- Преобразование 'x' из символа в число
34      mov eax,x
35      call atoi ; Вызов подпрограммы перевода символа в число
36      mov [x],eax ; запись преобразованного числа в 'x'
37
38
39
40 ; ----- Сравниваем 'a' и 'x' (как символы)
41      mov ecx,[x]
42      cmp [a],ecx; Сравниваем 'C' и 'A'
43      je check_B; если 'a=x', то переход на метку 'check_a'
44
45      mov eax,[x]; иначе 'ecx = C' a+x
46      mov ecx,[a]
47      add eax,ecx;
48      mov [min],eax ; 'min = C'
49      jmp fin
50
51 ; -----Преобразование 'min(A,C)' из символа в число
52 check_B:
53      mov eax,[a]
54      mov ecx,6
55      mul ecx
56      mov [min],eax;
57
58
59      ; -----Вывод результата
60 fin:
61      mov eax, msg2
62      call sprint ; Вывод сообщения 'Наибольшее число: '
63      mov eax,[min]
64      call iprintf; Вывод 'min(A,B,C)'
65      call quit; Выход
66
67
68

```

(рис. 2.18)

```

user@daborovikov:~/work/arch-pc/lab08$ ./lab8-4
Введите a: 2
Введите x: 1
Ответ: 3
user@daborovikov:~/work/arch-pc/lab08$ ./lab8-4
Введите a: 1
Введите x: 1
Ответ: 6
user@daborovikov:~/work/arch-pc/lab08$

```

Рис. 2.18: Проверка работы программы

Листинг программы:

%include 'in\_out.asm'

section .data

msg1 db 'Введите a: ',0h

```
msg2 db "Ответ: ",0h
```

```
msg3 db 'Введите x: ',0h
```

```
section .bss
```

```
min resb 10
```

```
x resb 10
```

```
a 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 ; запись преобразованного числа в 'a'
```

```
;----- Вывод сообщения 'Введите x:'
```

```
mov eax,msg3
```

```
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 ; запись преобразованного числа в 'x'
```

```
;----- Сравниваем 'a' и 'x' (как символы)
```

```
mov ecx,[x]
```

cmp [a],ecx; Сравниваем 'C' и 'A'

je check\_B; если 'a=x', то переход на метку 'check\_a'

mov eax,[x]; иначе 'ecx = C' a+x

mov ecx,[a]

add eax,ecx;

mov [min],eax ; 'min = C'

jmp fin

; ———Преобразование 'min(A,C)' из символа в число

check\_B:

mov eax,[a]

mov ecx,6

mul ecx

mov [min],eax;

; -----Вывод результата

fin: mov eax, msg2

call sprint ; Вывод сообщения 'Наибольшее число: '

```
mov eax,[min]
```

```
call iprintLF; Вывод 'min(A,B,C)'
```

```
call quit; Выход
```

## 3 Выводы

В ходе лабораторной работы мы изучили команды условного и безусловного переходов, приобрели навыки написания программ с использованием переходов, познакомились с назначением и структурой файла листинга.

[https://github.com/daBorovikov/study\\_2022-2023\\_arh-pc-](https://github.com/daBorovikov/study_2022-2023_arh-pc-)

...