

# **Отчёта по лабораторной работе 7**

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

Останин Владислав Александрович Мбв-01-21

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	21

## Список иллюстраций

2.1	Файл lab7-1.asm . . . . .	7
2.2	Программа lab7-1.asm . . . . .	8
2.3	Файл lab7-1.asm: . . . . .	9
2.4	Программа lab7-1.asm: . . . . .	10
2.5	Файл lab7-1.asm . . . . .	11
2.6	Программа lab7-1.asm . . . . .	12
2.7	Файл lab7-2.asm . . . . .	13
2.8	Программа lab7-2.asm . . . . .	13
2.9	Файл листинга lab7-2 . . . . .	14
2.10	ошибка трансляции lab7-2 . . . . .	15
2.11	файл листинга с ошибкой lab7-2 . . . . .	16
2.12	Файл lab7-3.asm . . . . .	17
2.13	Программа lab7-3.asm . . . . .	18
2.14	Файл lab7-4.asm . . . . .	19
2.15	Программа lab7-4.asm . . . . .	20

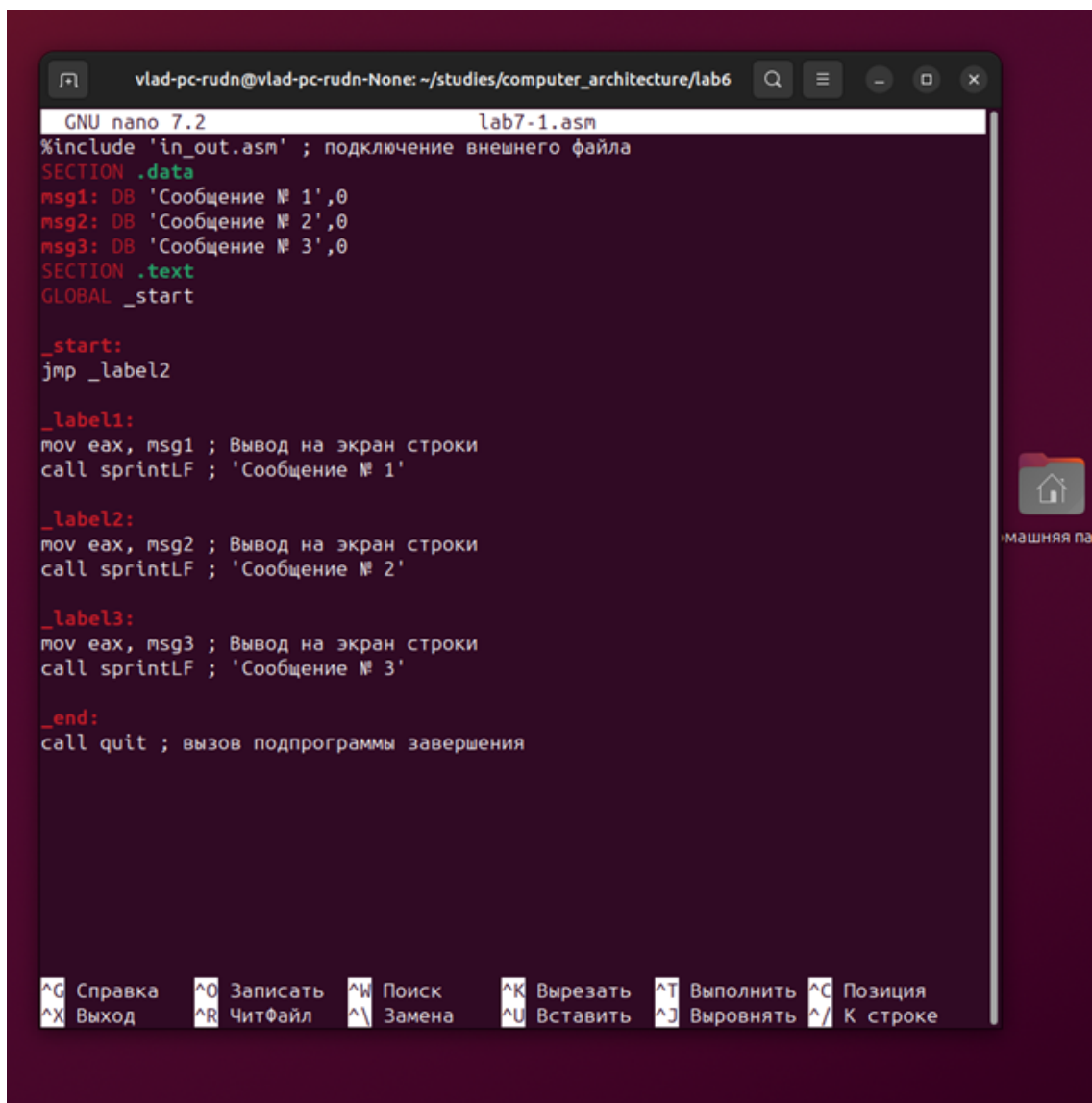
## **Список таблиц**

# 1 Цель работы

Целью работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

## 2 Выполнение лабораторной работы

1. Создайте каталог для программ лабораторной работы № 7, перейдите в него и создайте файл lab7-1.asm
2. 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.



```
GNU nano 7.2 lab7-1.asm
#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'

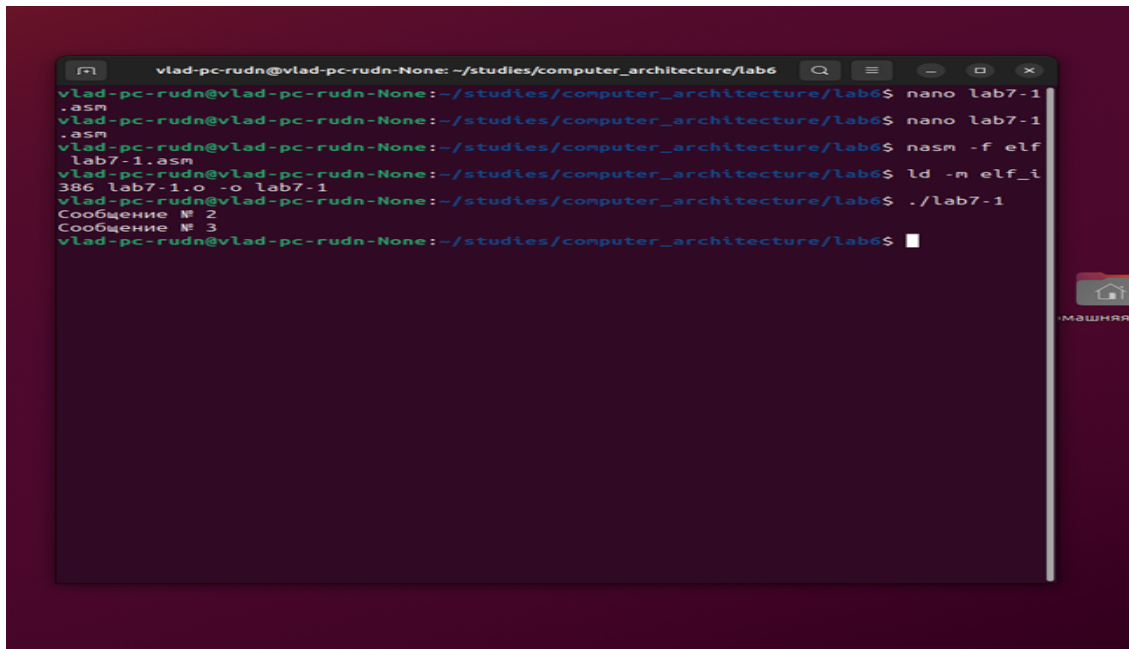
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'

_end:
call quit ; вызов подпрограммы завершения
```

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

Рис. 2.1: Файл lab7-1.asm

Создайте исполняемый файл и запустите его.

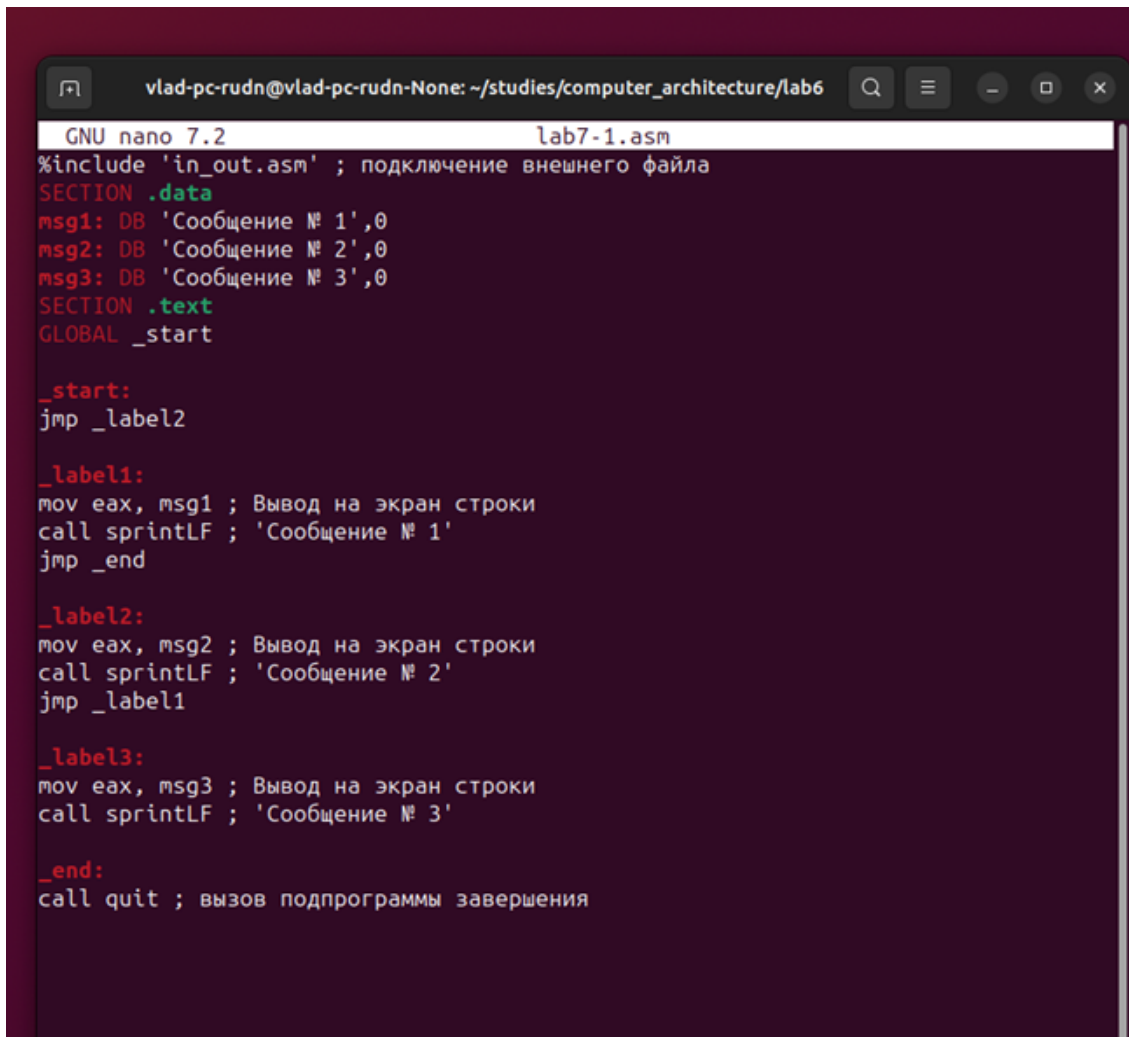


```
vlad-pc-rudn@vlad-pc-rudn-None: ~/studies/computer_architecture/lab6
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ nano lab7-1
. asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ nano lab7-1
. asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ nasm -f elf
lab7-1.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ ld -m elf_i386
386 lab7-1.o -o lab7-1
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ ./lab7-1
Сообщение № 2
Сообщение № 3
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$
```

Рис. 2.2: Программа lab7-1.asm

Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`). Измените текст программы в соответствии с листингом 7.2.





```
GNU nano 7.2 lab7-1.asm
%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'
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'

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 2.3: Файл lab7-1.asm:

```
.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ clenasm -f
elf lab7-1.asm^C
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ nasm -f elf
lab7-1.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ ld -m elf_i
386 lab7-1.o -o lab7-1
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ ./lab7-1
Сообщение № 2
Сообщение № 1
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$
```

Рис. 2.4: Программа lab7-1.asm:

Измените текст программы добавив или изменив инструкции `jmp`, чтобы вывод программы был следующим:

Сообщение № 3

Сообщение № 2

Сообщение № 1

```
GNU nano 7.2 lab7-1.asm
#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:
call quit ; вызов подпрограммы завершения
```

[ Прочитано 28 строк ]

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

Рис. 2.5: Файл lab7-1.asm

```
Сообщение № 2
Сообщение № 1
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ nano lab7-1
.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ nasm -f elf
lab7-1.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ ld -m elf_i
386 lab7-1.o -o lab7-1
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$
```

Рис. 2.6: Программа lab7-1.asm

3. Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры. Создайте исполняемый файл и проверьте его работу для разных значений В.

```

GNU nano 7.2 lab7-2.asm *
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'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход

```

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

Рис. 2.7: Файл lab7-2.asm

```

lab7-2.asm:17: error: invalid combination of opcode and operands
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ rm lab7-2.a
sm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ nasm -f elf
lab7-2.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ ld -m elf_i
386 lab7-2.o -o lab7-2
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ ./lab7-2
Введите B: 10
Наибольшее число: 50
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$

```

Рис. 2.8: Программа lab7-2.asm

4. Обычно nasm создаёт в результате ассемблирования только объектный

файл. Получить файл листинга можно, указав ключ -l и задав имя файла листинга в командной строке. Создайте файл листинга для программы из файла lab7-2.asm

```

108      <1>
109      <1> ;----- iprintf -----
110      <1> ; функция вывода на экран чисел в формате ASCII
111      <1> ; входные данные: mov eax,<int>
112      <1> iprintf:
113      00000086 E8C9FFFFFF      <1>      call     iprint
114      <1>
115      0000008B 50      <1>      push    eax
116      0000008C B80A000000      <1>      mov     eax, 0Ah
117      00000091 50      <1>      push    eax
118      00000092 89E0      <1>      mov     eax, esp
119      00000094 E876FFFFFF      <1>      call     sprint
120      00000099 58      <1>      pop     eax
121      0000009A 58      <1>      pop     eax
122      0000009B C3      <1>      ret
123      <1>
124      <1> ;----- atoi -----
125      <1> ; функция преобразования ascii-код символа в целое число
126      <1> ; входные данные: mov eax,<int>
127      <1> atoi:
128      0000009C 53      <1>      push    ebx
129      0000009D 51      <1>      push    ecx
130      0000009E 52      <1>      push    edx
131      0000009F 56      <1>      push    esi
132      000000A0 89C6      <1>      mov     esi, eax
133      000000A2 B800000000      <1>      mov     eax, 0
134      000000A7 B900000000      <1>      mov     ecx, 0
135      <1>
136      <1> .multiplyLoop:
137      000000AC 31DB      <1>      xor     ebx, ebx

```

Рис. 2.9: Файл листинга lab7-2

Внимательно ознакомиться с его форматом и содержимым. Подробно объяснить содержимое трёх строк файла листинга по выбору.

строка 128

- 128 - номер строки
- 0000009C - адрес
- 53 - машинный код
- push ebx - код программы

строка 129

- 129 - номер строки
- 0000009D - адрес
- 51 - машинный код
- push ecx- код программы

строка 130

- 130 - номер строки
- 0000009E - адрес
- 52 - машинный код
- push edx - код программы

Откройте файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалите один операнд. Выполните трансляцию с получением файла листинга

```
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ ./lab7-2
Введите B: 10
Наибольшее число: 50
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ nasm -f elf
lab7-2.asm
lab7-2.asm:17: error: invalid combination of opcode and operands
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$
```

Рис. 2.10: ошибка трансляции lab7-2

```

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 000000E8 B8[00000000]      mov eax,msg1
15 000000ED E81DFFFFFF        call sprint
16                               ; ----- Ввод 'B'
17                               mov ecx,
17                               *****
17                               error: invalid combination of opcode and operands
18 000000F2 BA0A000000        mov edx,10
19 000000F7 E847FFFFFF        call sread
20                               ; ----- Преобразование 'B' из символа в число
21 000000FC B8[0A000000]      mov eax,B
22 00000101 E896FFFFFF        call atoi ; Вызов подпрограммы перевода символа в число
23 00000106 A3[0A000000]      mov [B],eax ; запись преобразованного числа в 'B'
24                               ; ----- Записываем 'A' в переменную 'max'
25 0000010B 8B0D[35000000]    mov ecx,[A] ; 'ecx = A'
26 00000111 890D[00000000]    mov [max],ecx ; 'max = A'
27                               ; ----- Сравниваем 'A' и 'C' (как символы)
28 00000117 3B0D[39000000]    cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 0000011D 7F0C              jg check_B ; если 'A>C', то переход на метку 'check_B',
30 0000011F 8B0D[39000000]    mov ecx,[C] ; иначе 'ecx = C'

```

Рис. 2.11: файл листинга с ошибкой lab7-2

Объектный файл не смог создаться из-за ошибки. Но получился листинг, где выделено место ошибки.

5. Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу

для варианта 1 - 17, 23, 45



```
Открыть ▾ [F] lab7-3.asm ~/studies/computer_architecture/lab6
lab5-1.asm lab7-2.asm lab7-3.asm x

#include "in_out.asm"
SECTION .data
msgA: DB "Input A: ",0
msgB: DB "Input B: ",0
msgC: DB "Input C: ",0
answer: DB "Result: ",0

SECTION .bss
A: RESB 80
B: RESB 80
C: RESB 80
result: RESB 80
min: RESB 80

SECTION .text
GLOBAL _start

_start:
    mov eax,msgA
    call sprintf
    mov ecx,A
    mov edx,80
    call sread
    mov eax,A
    call atol
    mov [A],eax

    mov eax,msgB
    call sprintf
    mov ecx,B
    mov edx,80
    call sread
    mov eax,B
    call atol
    mov [B],eax

    mov eax,msgC
    call sprintf
    mov ecx,C
    mov edx,80
    call sread
    mov eax,C
    call atol
    mov [C],eax

;_____algorithm_____

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

    cmp ecx,[B] ; A<B
    jl check_C ; if a<b: goto check_C
    mov ecx,[B]
    mov [min],ecx ;else min = B

check_C:
    cmp ecx,[C]
    jl finish
    mov ecx,[C]
    mov [min],ecx

finish:
    mov eax,answer
    call sprintf

    mov eax,[min]
    call sprintf
    call quit
```

Рис. 2.12: Файл lab7-3.asm

```

lab7-2.asm:17: error: invalid combination of opcode and operands
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ nano lab7-3
;
mov     .asm
call    .asm
mov     ;
;
mov     vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ nasm -f elf
lab7-3.asm
mov     vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ ld -m elf_i
386 lab7-3.o -o lab7-3
mov     vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ ./lab7-3
Input A: 17
Input B: 23
Input C:
Smallest: 0
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ ./lab7-3
Input A: 17
Input B: 23
Input C: 45
Smallest: 17
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$

```

Рис. 2.13: Программа lab7-3.asm

6. 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.

для варианта 1

$$\begin{cases} 2a - x, & x < a \\ 8, & x \geq a \end{cases}$$

```
Открыть ▾
lab5-1.asm lab7-2.asm lab7-3.asm lab7-4.asm x
~/studies/computer_architecture/lab6
#include "io_out.asm"
SECTION .data
msgA: DB "Input A: ",0
msgX: DB "Input X: ",0
SECTION .bss
A: RESB 80
X: RESB 80
result: 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
mov [X],eax
;-----algorithm-----
mov ebx,[X]
mov edx,[A]
cmp ebx,edx
jb first
jmp second
first:
mov eax,[A]
mov ebx,2
mul ebx
sub eax,[X]
call iprintf
call quit
second:
mov eax,8
call iprintf
call quit
sd-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ nano lab7-
```

Рис. 2.14: Файл lab7-4.asm

```
Input B: 23
Input C: 45
Smallest: 17
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ nano lab7-4
.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ nasm -f elf
lab7-4.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ ld -m elf_i
386 lab7-4.o -o lab7-4
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$ ./lab7-4
Input A: 2
Input X: 1
3
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab6$
```

Рис. 2.15: Программа lab7-4.asm

## 3 Выводы

Изучили команды условного и безусловного переходов, познакомились с фалом листинга.