

Отчёта по лабораторной работе №7

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

Спелов Андрей Николаевич

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	11
3.3	Задание для самостоятельной работы	14
4	Выводы	20

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	7
3.3	Запускаем файл и смотрим на его работу	7
3.4	Изменяем файл	8
3.5	Запускаем файл и смотрим на его работу	8
3.6	Редактируем файл	9
3.7	Проверяем, сошелся ли наш вывод с данным в условии выводом .	9
3.8	Создаем файл командой <code>touch</code>	9
3.9	Заполняем файл	10
3.10	Смотрим на работу программ	10
3.11	Создаем файл листинга	11
3.12	Изучаем файл	11
3.13	Удаляем операндум из файла	13
3.14	Транслируем файл	14
3.15	Изучаем файл с ошибкой	14
3.16	Создаем файл командой <code>touch</code>	15
3.17	Пишем программу	16
3.18	Смотрим на работу программы(всё верно)	16
3.19	Создаем файл командой <code>touch</code>	17
3.20	Пишем программу	18
3.21	Проверяем работу программы	18
3.22	Проверяем работу программы	19

1 Цель работы

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

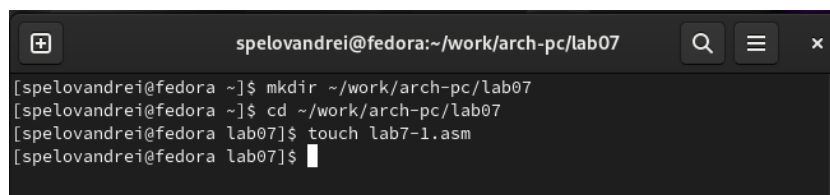
2 Задание

Написать программы для решения системы выражений.

3 Выполнение лабораторной работы

3.1 Реализация переходов в NASM

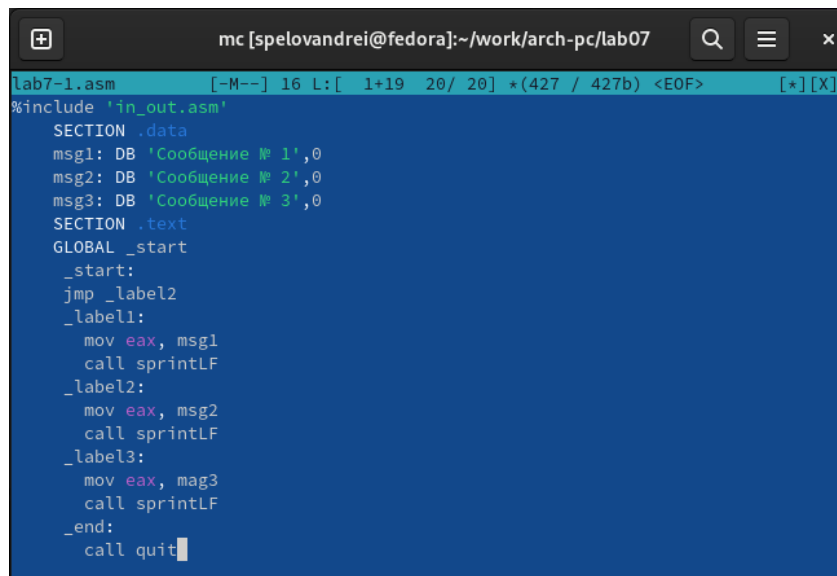
Создаем каталог для программ ЛБ7, и в нем создаем файл (рис. 3.1).

A terminal window with a dark background. The title bar shows the user 'spelovandrei@fedora' and the current directory '~/work/arch-pc/lab07'. The terminal contains the following commands and their outputs:

```
[spelovandrei@fedora ~]$ mkdir ~/work/arch-pc/lab07
[spelovandrei@fedora ~]$ cd ~/work/arch-pc/lab07
[spelovandrei@fedora lab07]$ touch lab7-1.asm
[spelovandrei@fedora lab07]$
```

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

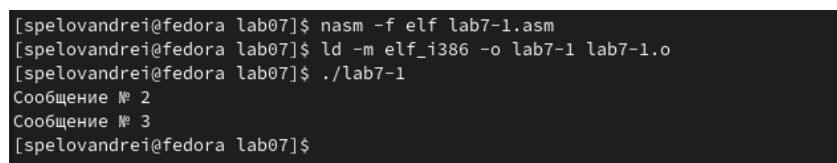
Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 7.1 (рис. 3.2).



```
lab7-1.asm [-M--] 16 L: [ 1+19 20/ 20] *(427 / 427b) <EOF> [*][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 sprintLF
_label2:
mov eax, msg2
call sprintLF
_label3:
mov eax, msg3
call sprintLF
_end:
call quit
```

Рис. 3.2: Заполняем файл

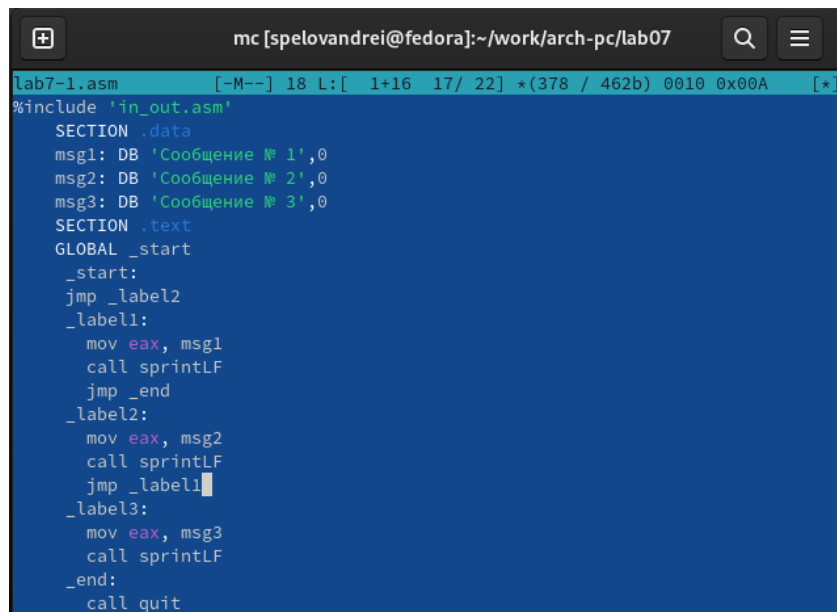
Создаем исполняемый файл и запускаем его (рис. 3.3).



```
[spelovandrei@fedora lab07]$ nasm -f elf lab7-1.asm
[spelovandrei@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[spelovandrei@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 3
[spelovandrei@fedora lab07]$
```

Рис. 3.3: Запускаем файл и смотрим на его работу

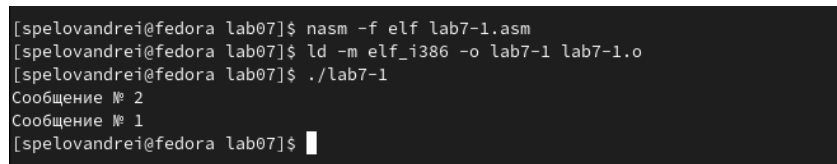
Снова открываем файл для редактирования и изменяем его в соответствии с листингом 7.2 (рис. 3.4).



```
lab7-1.asm [-M--] 18 L:[ 1+16 17/ 22] *(378 / 462b) 0010 0x00A [*]
#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
```

Рис. 3.4: Изменяем файл

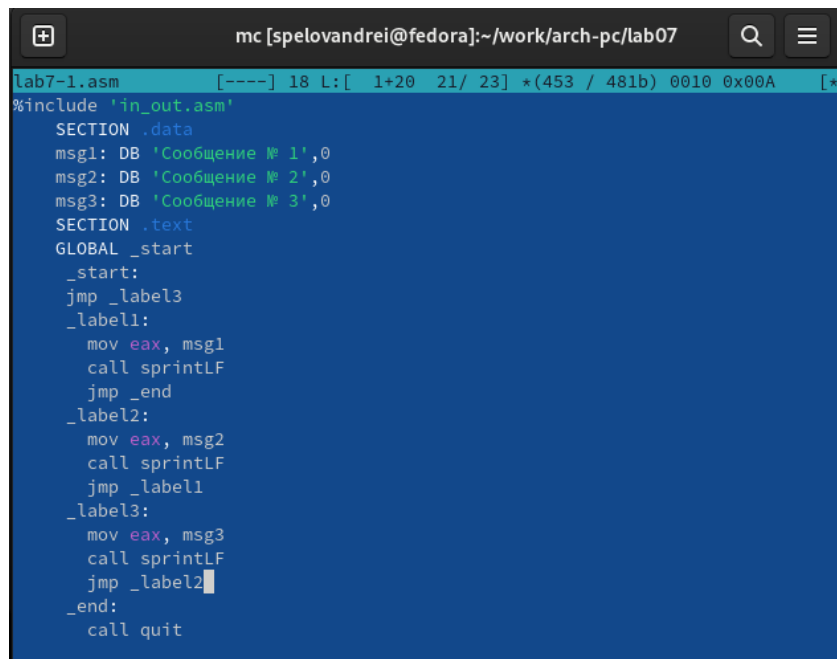
Создаем исполняемый файл и запускаем его (рис. 3.5).



```
[spelovandrei@fedora lab07]$ nasm -f elf lab7-1.asm
[spelovandrei@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[spelovandrei@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 1
[spelovandrei@fedora lab07]$
```

Рис. 3.5: Запускаем файл и смотрим на его работу

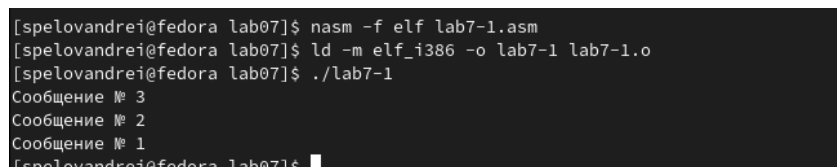
Снова открываем файл для редактирования и изменяем его, чтобы произошел данный вывод (рис. 3.6).



```
lab7-1.asm [----] 18 L: [ 1+20 21/ 23] *(453 / 481b) 0010 0x00A [*]
%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 sprintf
jmp _end
_label2:
mov eax, msg2
call sprintf
jmp _label1
_label3:
mov eax, msg3
call sprintf
jmp _label2
_end:
call quit
```

Рис. 3.6: Редактируем файл

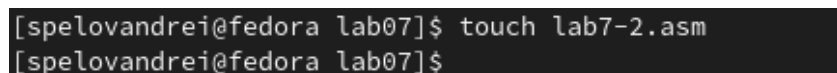
Создаем исполняемый файл и запускаем его (рис. 3.7).



```
[spelovandrei@fedora lab07]$ nasm -f elf lab7-1.asm
[spelovandrei@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[spelovandrei@fedora lab07]$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[spelovandrei@fedora lab07]$
```

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

Создаем новый файл (рис. 3.8).



```
[spelovandrei@fedora lab07]$ touch lab7-2.asm
[spelovandrei@fedora lab07]$
```

Рис. 3.8: Создаем файл командой touch

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

```
mc [spelovandrei@fedora]:~/work/arch-pc/lab07
lab7-2.asm [----] 12 L: [ 1+ 2 3/ 37] *(55 / 562b) 0039 0x027
#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
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
```

Рис. 3.9: Заполняем файл

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

```
[spelovandrei@fedora lab07]$ nasm -f elf lab7-2.asm
[spelovandrei@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[spelovandrei@fedora lab07]$ ./lab7-2
5
20
[spelovandrei@fedora lab07]$ nasm -f elf lab7-2.asm
[spelovandrei@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[spelovandrei@fedora lab07]$ ./lab7-2
10
20
[spelovandrei@fedora lab07]$ nasm -f elf lab7-2.asm
[spelovandrei@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[spelovandrei@fedora lab07]$ ./lab7-2
1
20
[spelovandrei@fedora lab07]$
```

Рис. 3.10: Смотрим на работу программ

3.2 Изучение структуры файлы листинга

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

```
[spelovandrei@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm  
[spelovandrei@fedora lab07]$
```

Рис. 3.11: Создаем файл листинга

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

```
lab7-2.lst  [----]  0 L: [ 1+ 0 1/208]  *(0 /12617b) 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          00000003 803B00      <1> nextchar:-----  
9          00000006 7403        <1> cmp   byte [eax], 0...  
10         00000008 40          <1> jz    finished.....  
11         00000009 EBF8        <1> inc   eax.....  
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  
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 E8E8FFFFFF   <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 BB01000000   <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.....  
1.Помощь 2.Сохранить 3.Блок 4.Замена 5.Копия 6.Пере-ить 7.Поиск 8.Удалить 9.МенюМС 10.Выход
```

Рис. 3.12: Изучаем файл

Строка 33: 0000001D-адрес в сегменте кода, BB01000000-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. 3.13).

```

lab7-2.asm [-M--] 11
#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
    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

```

Рис. 3.13: Удаляем операндум из файла

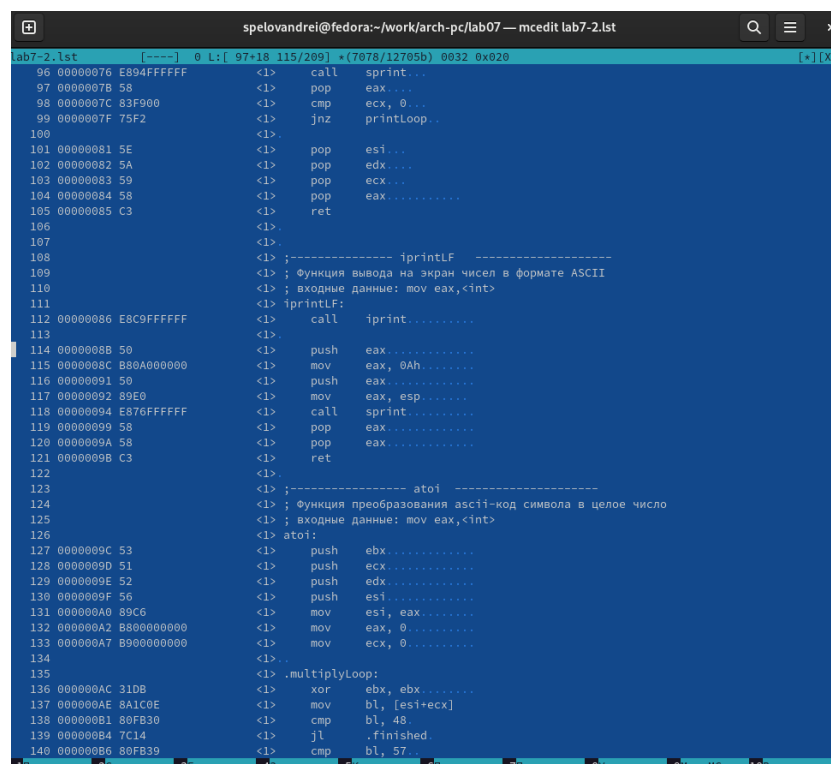
Транслируем с получением файла листинга (рис. 3.14).

```
[spelovandrei@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:16: error: invalid combination of opcode and operands
[spelovandrei@fedora lab07]$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2  lab7-2.asm  lab7-2.lst
[spelovandrei@fedora 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. 3.15).



```
lab7-2.lst 0 L: [ 97*16 115/209] * (7076/12705b) 0032 0x020
96 00000076 E894FFFFFF <I> call sprint...
97 00000078 58 <I> pop eax...
98 0000007C 83F900 <I> cmp ecx, 0...
99 0000007F 75F2 <I> jnz printLoop...
100 <I>
101 00000081 5E <I> pop esi...
102 00000082 5A <I> pop edx...
103 00000083 59 <I> pop ecx...
104 00000084 58 <I> pop eax...
105 00000085 C3 <I> ret
106 <I>
107 <I>
108 <I> ;----- iprintf -----
109 <I> ; функция вывода на экран чисел в формате ASCII
110 <I> ; входные данные: mov eax,<int>
111 <I> iprintf:
112 00000086 E8C9FFFFFF <I> call iprint...
113 <I>
114 0000008B 50 <I> push eax...
115 0000008C B8A0000000 <I> mov eax, 0Ah...
116 00000091 50 <I> push eax...
117 00000092 89E0 <I> mov eax, esp...
118 00000094 E876FFFFFF <I> call sprint...
119 00000099 58 <I> pop eax...
120 0000009A 58 <I> pop eax...
121 0000009B C3 <I> ret
122 <I>
123 <I> ;----- atoi -----
124 <I> ; функция преобразования ascii-код символа в целое число
125 <I> ; входные данные: mov eax,<int>
126 <I> atoi:
127 0000009C 53 <I> push ebx...
128 0000009D 51 <I> push ecx...
129 0000009E 52 <I> push edx...
130 0000009F 56 <I> push esi...
131 000000A0 89C6 <I> mov esi, eax...
132 000000A2 B800000000 <I> mov eax, 0...
133 000000A7 B900000000 <I> mov ecx, 0...
134 <I>
135 <I> .multiplyLoop:
136 000000AC 31DB <I> xor ebx, ebx...
137 000000AE 8A1C0E <I> mov bl, [esi+ecx]
138 000000B1 80FB30 <I> cmp bl, 48...
139 000000B4 7C14 <I> jl .finished...
140 000000B6 80FB39 <I> cmp bl, 57...
```

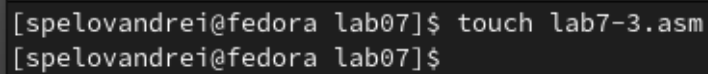
Рис. 3.15: Изучаем файл с ошибкой

3.3 Задание для самостоятельной работы

ВАРИАНТ-20

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных x, y и 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. 3.16).



```
[spelovandrei@fedora lab07]$ touch lab7-3.asm  
[spelovandrei@fedora lab07]$
```

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. 3.17).

```

mc [spelovandrei@fedora]:~/work/arch-pc/lab07
lab7-3.asm [-M--] 12 L:[ 1+40 41/ 41] *(674 / 674b) <EOF>
#include 'in_out.asm'
section .data
    msg1 db 'Введите B: ',0h
    msg2 db "Наименьшее число: ",0h
    A dd '95'
    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 iprintLF
    call quit

```

Рис. 3.17: Пишем программу

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

```

[spelovandrei@fedora lab07]$ nasm -f elf lab7-3.asm
[spelovandrei@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[spelovandrei@fedora lab07]$ ./lab7-3
Введите B: 2
Наименьшее число: 2
[spelovandrei@fedora lab07]$

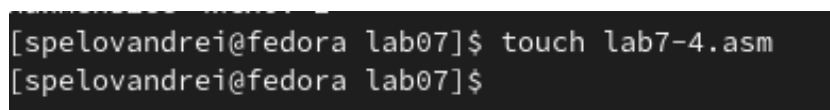
```

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 вы-

числений. Вид функции $\times(\times)$ 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 \times и \times из 7.6.

Создаем новый файл (рис. 3.19).



```
[spelovandrei@fedora lab07]$ touch lab7-4.asm  
[spelovandrei@fedora lab07]$
```

Рис. 3.19: Создаем файл командой touch

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

```
mc [spelovandrei@fedora]:~/work/arch-pc/lab07
lab7-4.asm [----] 12 L: [ 1+41 42/ 44] *(644 / 679b) 0091 0x058
#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,[a]
    mov [res],ecx
    jmp fin
check_A:
    mov ecx,5
    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 $x=1$ и $a=2$ (рис. 3.21).

```
[spelovandrei@fedora lab07]$ nasm -f elf lab7-4.asm
[spelovandrei@fedora lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o
[spelovandrei@fedora lab07]$ ./lab7-4
Введите x: 1
Введите a: 2
F(x) = 5
[spelovandrei@fedora lab07]$
```

Рис. 3.21: Проверяем работу программы

Транслируем файл и проверяем его работу при $x=2$ и $a=1$ (рис. 3.22).

```
[spelovandrei@fedora lab07]$ nasm -f elf lab7-4.asm
[spelovandrei@fedora lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o
[spelovandrei@fedora lab07]$ ./lab7-4
Введите x: 2
Введите a: 1
F(x) = 1
```

Рис. 3.22: Проверяем работу программы

4 Выводы

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