

Отчёта по лабораторной работе №7

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

Гурбанов Сарча

Содержание

1	<u>Цель работы</u>	4
2	<u>Задание</u>	5
3	<u>Выполнение лабораторной работы</u>	6
1.	Реализация переходов в NASM	6
2.	Изучение структуры файлы листинга	11
3.	Задание для самостоятельной работы	14
4	<u>Выводы</u>	20

Список иллюстраций

1.	<u>Создаем каталог с помощью команды mkdir и файл с помощью команды touch</u>	6
2.	<u>Заполняем файл</u>	7
3.	<u>Запускаем файл и смотрим на его работу</u>	7
4.	<u>Изменяем файл</u>	8
5.	<u>Запускаем файл и смотрим на его работу</u>	8
6.	<u>Редактируем файл</u>	9
7.	<u>Проверяем, сошелся ли наш вывод с данным в условии выводом</u>	9
8.	<u>Создаем файл командой touch</u>	9
9.	<u>Заполняем файл</u>	10
10.	<u>Смотрим на работу программ</u>	10
11.	<u>Создаем файл листинга</u>	11
12.	<u>Изучаем файл</u>	11
13.	<u>Удаляем операндум из файла</u>	13
14.	<u>Транслируем файл</u>	14
15.	<u>Изучаем файл с ошибкой</u>	14
16.	<u>Создаем файл командой touch</u>	15
17.	<u>Пишем программу</u>	16
18.	<u>Смотрим на работу программы(всё верно)</u>	16
19.	<u>Создаем файл командой touch</u>	17
20.	<u>Пишем программу</u>	18
21.	<u>Проверяем работу программы</u>	18
22.	<u>Проверяем работу программы</u>	19

1 Цель работы

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

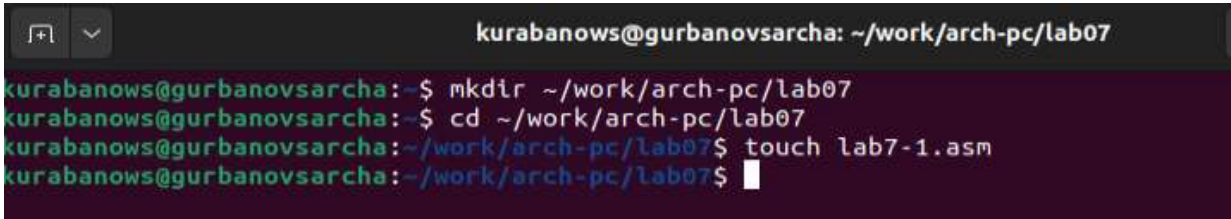
2 Задание

Написать программы для решения системы выражений.

3 Выполнение лабораторной работы

3.1 Реализация переходов в NASM

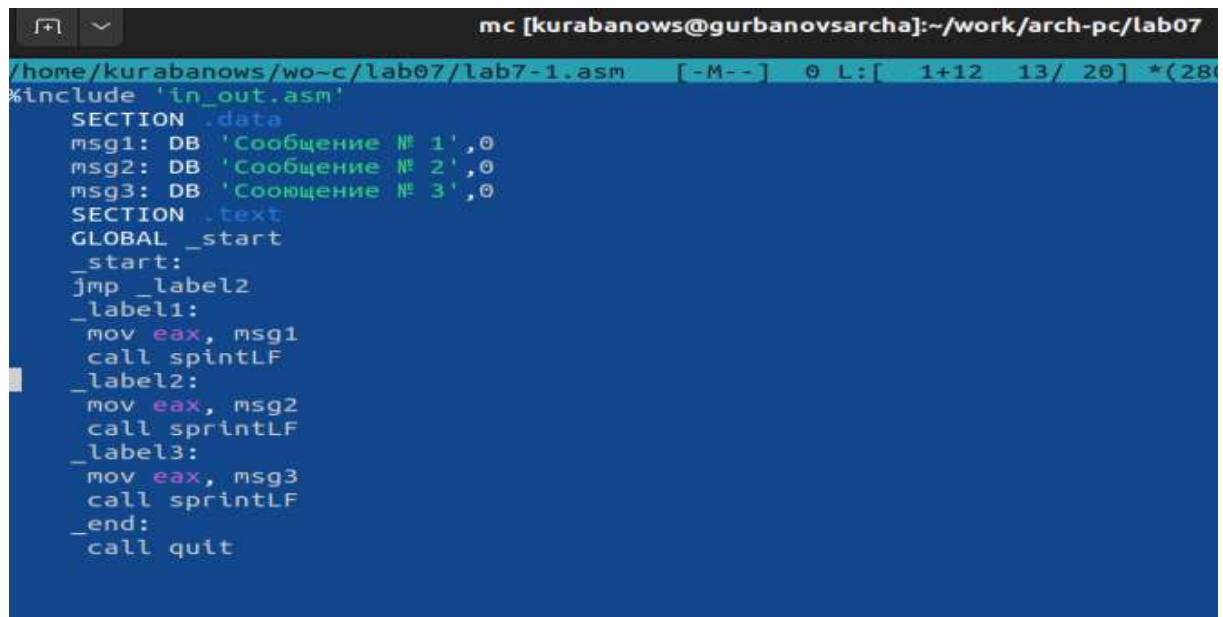
Создаем каталог для программ ЛБ7, и в нем создаем файл (рис. 3.1).



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

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

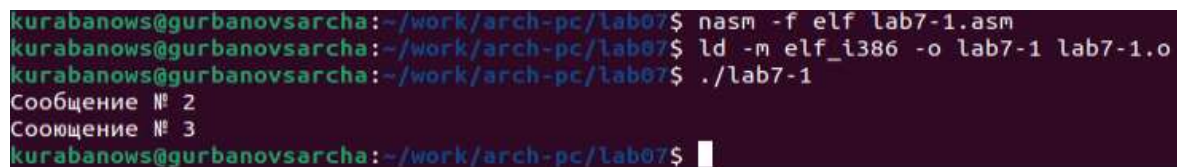
Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 7.1 (рис. 3.2).



```
mc [kurabanows@gurbanovsarcha]:~/work/arch-pc/lab07
/home/kurabanows/work/arch-pc/lab07/lab7-1.asm [-M--] 0 L: [ 1+12 13/ 20] *(28
%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 spintLF
_label2:
mov eax, msg2
call sprintLF
_label3:
mov eax, msg3
call sprintLF
_end:
call quit
```

Рис. 3.2: Заполняем файл

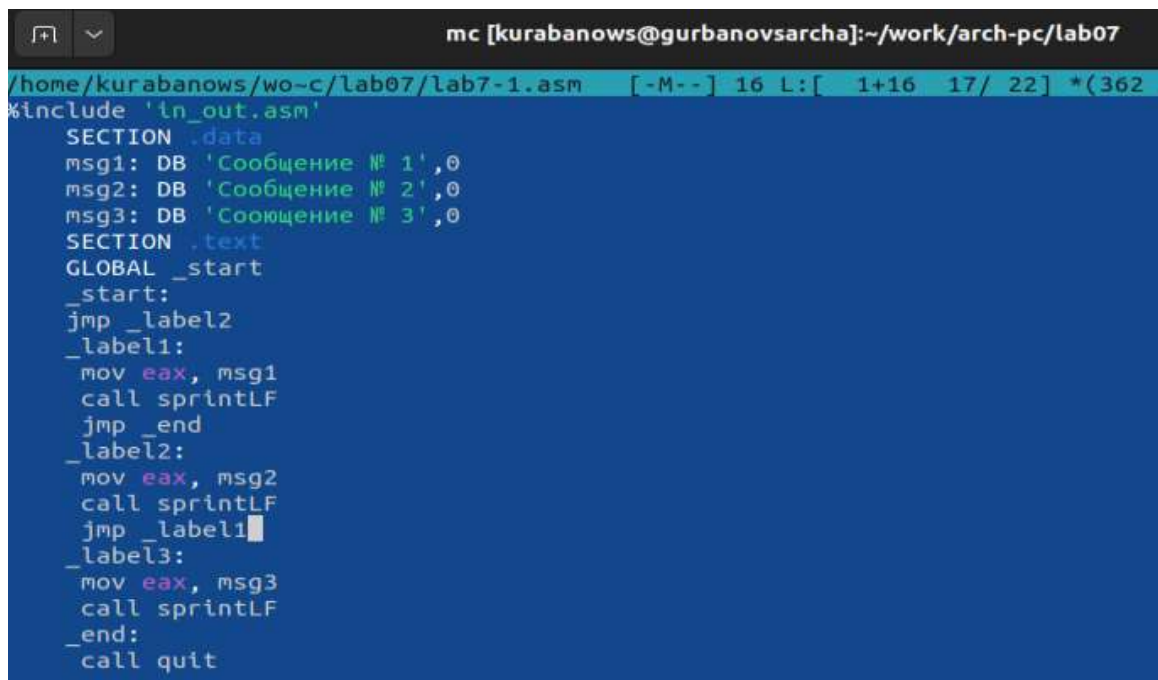
Создаем исполняемый файл и запускаем его (рис. 3.3).



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

Рис. 3.3: Запускаем файл и смотрим на его работу

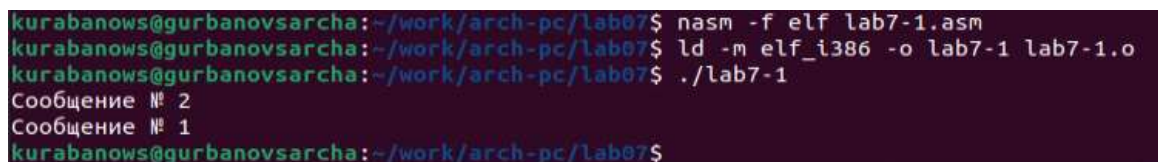
Снова открываем файл для редактирования и изменяем его в соответствии с листингом 7.2 (рис. 3.4).



```
mc [kurabanows@gurbanovsarcha]:~/work/arch-pc/lab07
/home/kurabanows/work/arch-pc/lab07/lab7-1.asm [-M--] 16 L:[ 1+16 17/ 22] *(362
%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

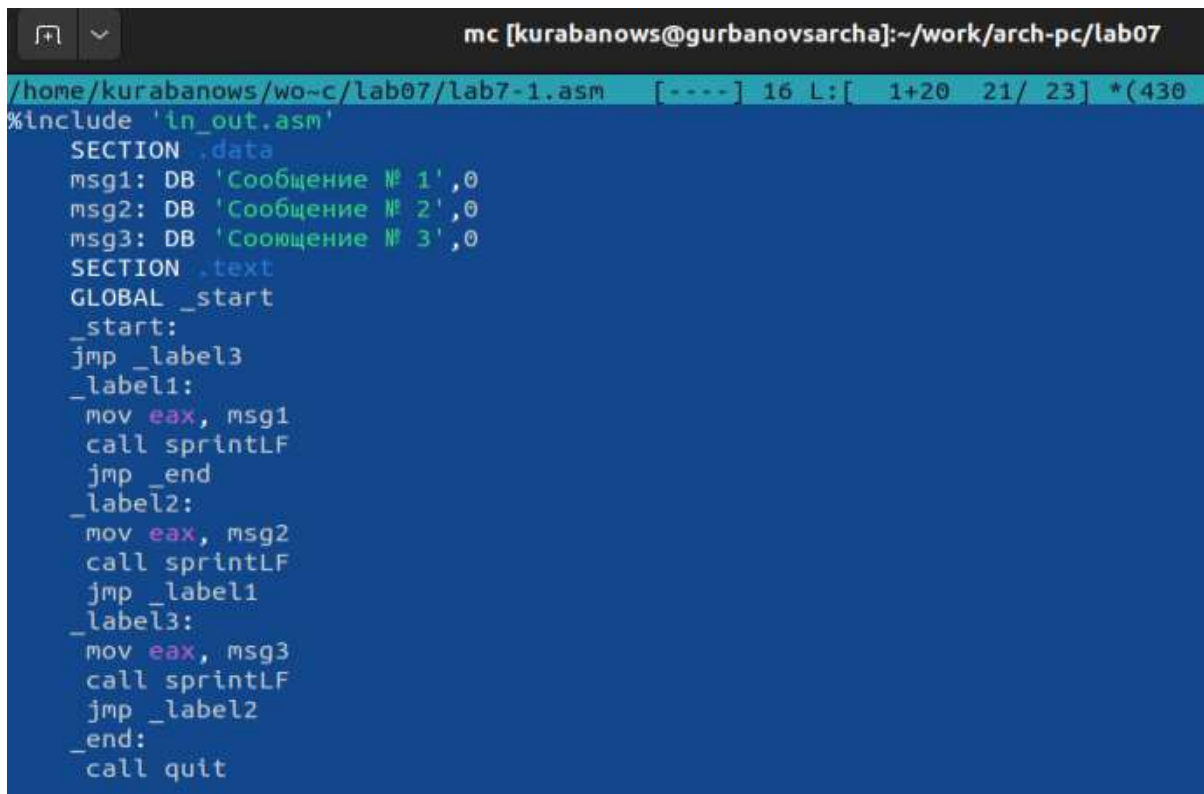
Создаем исполняемый файл и запускаем его (рис. 3.5).



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

Рис. 3.5: Запускаем файл и смотрим на его работу

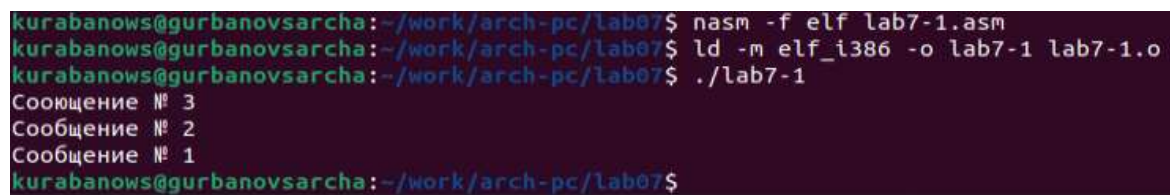
Снова открываем файл для редактирования и изменяем его, чтобы произошел данный вывод (рис. 3.6).



```
mc [kurabanows@gurbanovsarcha]:~/work/arch-pc/lab07
/home/kurabanows/work/arch-pc/lab07/lab7-1.asm [----] 16 L: [ 1+20 21/ 23] *(430
%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 файл

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



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

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

Создаем новый файл (рис. 3.8).



```
kurabanows@gurbanovsarcha:~/work/arch-pc/lab07$ touch lab7-2.asm
kurabanows@gurbanovsarcha:~/work/arch-pc/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 [kurabanows@gurbanovsarcha]:~/work/arch-pc/lab07
/home/kurabanows/work/arch-pc/lab07/lab7-2.asm [-M--] 16 L: [ 1+32 33/ 48] *(499
#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:
    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

    cmp ecx, [C]
    jg check_B
    mov ecx, [C]
    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 lprintLF
    call quit

```

Рис. 3.9: Заполняем файл

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

```

kurabanows@gurbanovsarcha:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
kurabanows@gurbanovsarcha:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
kurabanows@gurbanovsarcha:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наибольшее число: 50
kurabanows@gurbanovsarcha:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
kurabanows@gurbanovsarcha:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
kurabanows@gurbanovsarcha:~/work/arch-pc/lab07$

```

Рис. 3.10: Смотрим на работу программ

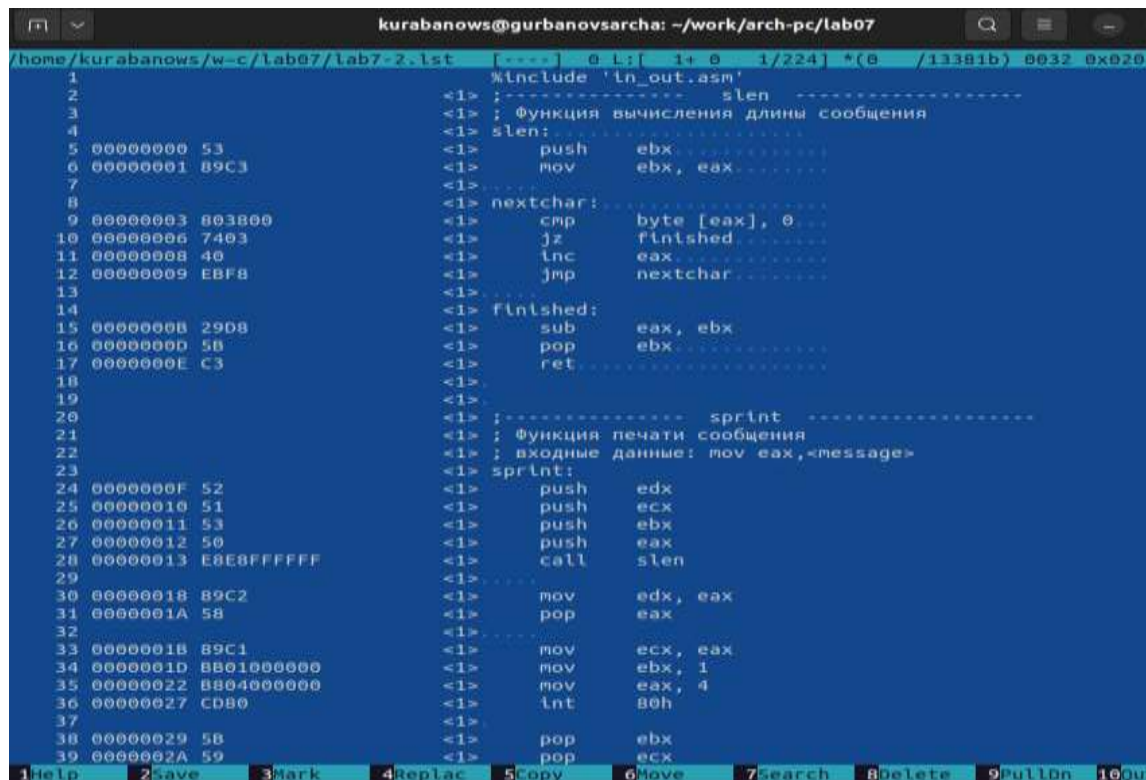
3.2 Изучение структуры файлы листинга

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

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

Рис. 3.11: Создаем файл листинга

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



```
1 %include 'in_out.asm'
2 ;----- slen -----
3 <1> ; Функция вычисления длины сообщения
4 <1> slen:
5 00000000 53 <1> push ebx
6 00000001 89C3 <1> mov ebx, eax
7
8 <1> nextchar:
9 00000003 803800 <1> cmp byte [eax], 0
10 00000006 7403 <1> jz finished
11 00000008 40 <1> inc eax
12 00000009 EBF8 <1> jmp nextchar
13
14 <1> finished:
15 0000000B 29D8 <1> sub eax, ebx
16 0000000D 5B <1> pop ebx
17 0000000E C3 <1> ret
18
19
20 <1> ;----- sprint -----
21 <1> ; Функция печати сообщения
22 <1> ; входные данные: mov eax, «message»
23 <1> sprint:
24 0000000F 52 <1> push edx
25 00000010 51 <1> push ecx
26 00000011 53 <1> push ebx
27 00000012 50 <1> push eax
28 00000013 E8E8FFFFFF <1> call slen
29
30 00000018 89C2 <1> mov edx, eax
31 0000001A 58 <1> pop eax
32
33 0000001B 89C1 <1> mov ecx, eax
34 0000001D BB01000000 <1> mov ebx, 1
35 00000022 B804000000 <1> mov eax, 4
36 00000027 CD80 <1> int 80h
37
38 00000029 5B <1> pop ebx
39 0000002A 59 <1> pop ecx
```

Рис. 3.12: Изучаем файл

Строка 33: 0000001D-адрес в сегменте кода, B801000000-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).

```

mc [kurabanows@gu
/home/kurabanows/wo~c/lab07/lab7-2.asm [-M-
#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:
    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

    cmp ecx, [C]
    jg check_B
    mov ecx, [C]
    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).

```
kurabanows@gurbanovsarcha:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:17: error: invalid combination of opcode and operands
kurabanows@gurbanovsarcha:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.lst
kurabanows@gurbanovsarcha:~/work/arch-pc/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).

```

C:\Program Files\Kaspersky Lab\Kaspersky Security Center\KSC\bin> kurabanows@gurbanovsarcha: ~/work/arch-pc/lab07
/home/kurabanows/w-c/lab07/lab7-2.lst 0 L: 954 0 95/225 *(5833/13468b) 0032 0x020
95 00000073 49 <1> dec ecx,ecx
96 00000074 89E0 <1> mov eax,esp
97 00000076 E894FFFFFF <1> call sprint
98 00000078 58 <1> pop eax
99 0000007C 83F900 <1> cmp ecx,0
100 0000007F 75F2 <1> jnz printLoop
101 <1>
102 00000081 5E <1> pop esi
103 00000082 5A <1> pop edx
104 00000083 59 <1> pop ecx
105 00000084 58 <1> pop eax
106 00000085 C3 <1> ret
107 <1>
108 <1>
109 <1> ;----- tprintlnf -----
110 <1> ; Функция вывода на экран чисел в формате ASCII
111 <1> ; входные данные: mov eax,<int>
112 <1> tprintlnf:
113 00000086 E8C9FFFFFF <1> call tprint
114 <1>
115 0000008B 50 <1> push eax
116 0000008C B80A000000 <1> mov eax,0Ah
117 00000091 50 <1> push eax
118 00000092 89E8 <1> mov ecx,esp
119 00000094 EB76FFFFFF <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 8900000000 <1> mov ecx,0
135 <1>
136 <1> multiplyLoop:
137 000000AC 31DB <1> xor ebx,ebx
138 000000AE 8A1C0E <1> mov bl,[esi+ecx]
139 000000B1 80FB30 <1> cmp bl,40
140 000000B4 7C14 <1> jl finished
141 000000B6 80FB39 <1> cmp bl,57
142 000000B9 7F0F <1> jg finished
143 <1>
144 000000BB 80FB30 <1> sub bl,40

```

Рис. 3.15: Изучаем файл с ошибкой

3.3 Задание для самостоятельной работы

ВАРИАНТ-20

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных ~~??~~ и с. 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).



```
kurabanows@gurbanovsarcha:~/work/arch-pc/lab07$ touch lab7-3.asm
kurabanows@gurbanovsarcha:~/work/arch-pc/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 [kurabanows@gurbanovsarcha]:~/work/arch-pc/lab07
/home/kurabanows/work/arch-pc/lab07/lab7-3.asm [-M--] 13 L: [ 1+38 39/ 39] *(664 /
%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]
    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).

```

kurabanows@gurbanovsarcha:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
kurabanows@gurbanovsarcha:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
kurabanows@gurbanovsarcha:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 2
Наименьшее число2
kurabanows@gurbanovsarcha:~/work/arch-pc/lab07$

```

Рис. 3.18: Смотрим на работу программы(всё верно)

2. Напишите программу, которая для введенных с клавиатуры значений \diamond и \diamond вычисляет значение заданной функции $\diamond(\diamond)$ и выводит результат вы-

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

Создаем новый файл (рис. 3.19).



```
kurabanows@gurbanovsarcha:~/work/arch-pc/lab07$ touch lab7-4.asm
kurabanows@gurbanovsarcha:~/work/arch-pc/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 [kurabanows@gurbanovsarcha]:~/work/arch-pc/lab07
/home/kurabanows/work/arch-pc/lab07/lab7-4.asm [-M--] 13 L: [ 1+43 44/ 44] *(72)
%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 atol
    mov [x], eax
    mov eax, msg2
    call sprint
    mov ecx, a
    mov edx, 80
    call sread
    mov eax, a
    call atol
    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 iptintLF
    call quit

```

Рис. 3.20: Пишем программу

Транслируем файл и проверяем его работу при $x=1$ и $a=2$ (рис. 3.21).

```

kurabanows@gurbanovsarcha:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
kurabanows@gurbanovsarcha:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
kurabanows@gurbanovsarcha:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 1
Введите a: 2
F(x) = 5
kurabanows@gurbanovsarcha:~/work/arch-pc/lab07$

```

Рис. 3.21: Проверяем работу программы

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

```
kurabanows@gurbanovsarcha:~/work/arch-pc/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.