

**Лаборная работа №8 Команды
безусловного и условного переходов в
Nasm. Программирование ветвлений.**

НММ-бд-02-22

Крухмалев Артём Владиславович

Содержание

1	Цель работы	3
2	Задание	4
3	Выполнение лабораторной работы	5
4	Самостоятельная работа	10
5	Выводы	12

1 Цель работы

Изучить команды условного и безусловного перехода, познакомиться с программой ветвлений

2 Задание

Написать программу с использованием ветвлений

3 Выполнение лабораторной работы

1. С помощью терминала создадим подкаталог, создадим файл lab8-1.asm

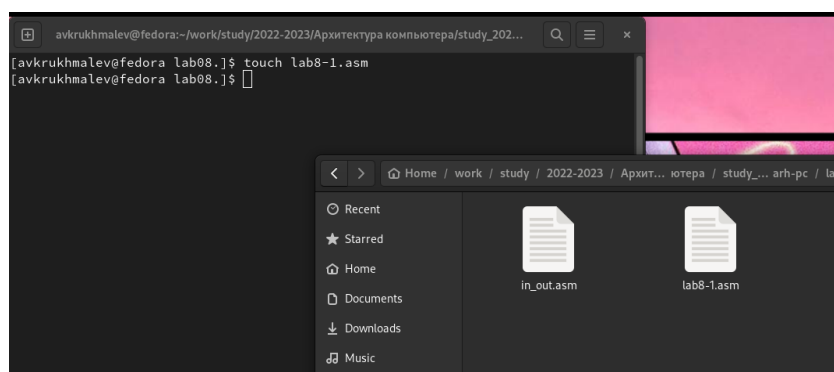


Рис. 3.1: Новый каталог

2. Изучим и запишем в него код из листинга, откомпилируем и запустим файл

```
[avkrukhmalev@fedora lab08.]$ nasm -f elf lab8-1.asm
[avkrukhmalev@fedora lab08.]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[avkrukhmalev@fedora lab08.]$ ./lab8-1
Сообщение №2
Сообщение №3
[avkrukhmalev@fedora lab08.]$
```

lab8-1.asm

~/work/study/2022-2023/Архитектур..._2022-2

Open

%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: Простейшая задача

3. Изменим код программы и посмотрим, что он нам выведет

```
[avkrukhmalev@fedora lab08.]$ nasm -f elf lab8-1.asm
[avkrukhmalev@fedora lab08.]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[avkrukhmalev@fedora lab08.]$ ./lab8-1
Сообщение №3
Сообщение №2
Сообщение №1
[avkrukhmalev@fedora lab08.]$
```

lab8-1.asm

~/work/study/2022-2023/Архитектур..._2022-2

Open

%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

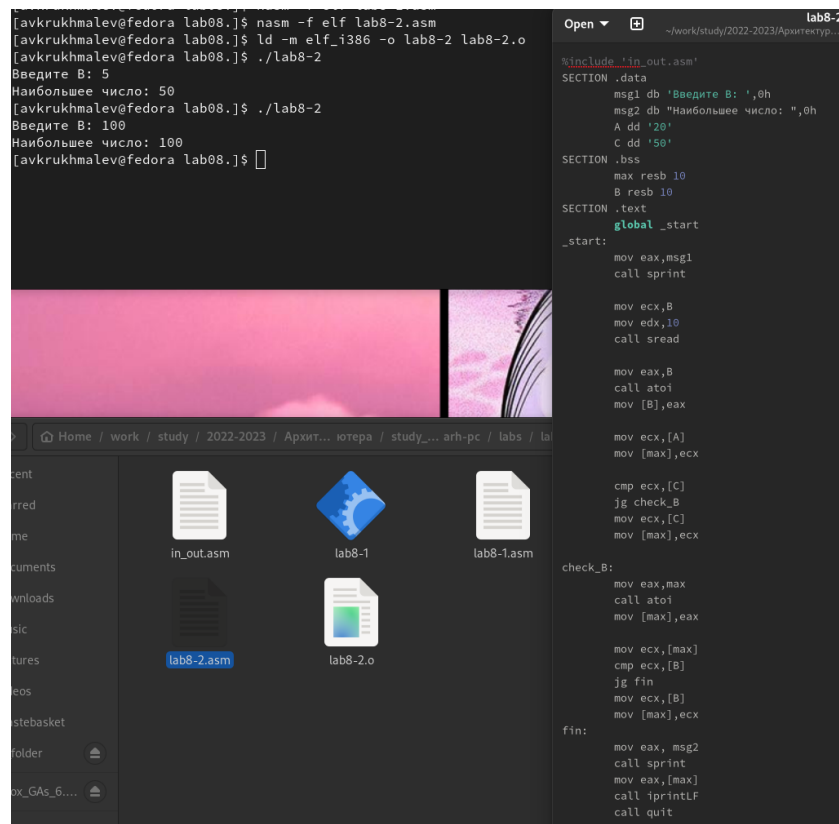
jmp _label2

_end:

call quit

Рис. 3.3: Измененный файл

4. Создадим новый файл, запишем в него предложенный код, предварительно изучив его, посмотрим, что он выводит нам при разных значениях



The screenshot shows a terminal window on the left and a code editor on the right. The terminal displays the following commands and output:

```
[avkrukhmalev@fedora lab08]$ nasm -f elf lab8-2.asm
[avkrukhmalev@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[avkrukhmalev@fedora lab08]$ ./lab8-2
Введите B: 5
Наибольшее число: 50
[avkrukhmalev@fedora lab08]$ ./lab8-2
Введите B: 100
Наибольшее число: 100
[avkrukhmalev@fedora lab08]$
```

The code editor shows the assembly code for lab8-2.asm:

```
%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 iprintlnLF
    call quit
```

Рис. 3.4: Нахождение максимального числа

5. С помощью команды `nasm -f elf -l lab8-2.lst lab8-2.asm` создадим файл листинга.

000000F2(строка на 16 месте-это её адрес); B90A000000 (машинный код); `mov ecx,C` (в регистр `eax` вносим значение `C`) 000000F7 (строка на 17 месте-это её адрес); BA0A000000 (машинный код); `mov edx,10` (в регистр `eax` вносим значение 10) 000000FC (строка на 18месте-это её адрес); E842FFFFFF (машинный код); `call sread` (Функция считывания сообщения)

```

lab8-2.lst  [----]  0 L: [ 1+ 0 1/224] * (0 /13262b) 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
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          <1>
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 E8E8FFFF    <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 B801000000   <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
40 0000002B 5A         <1> pop     edx
41 0000002C C3         <1> ret
42
43          <1>
44          <1> ;----- sprintf -----
45          <1> ; Функция печати сообщения с переводом строки
46          <1> ; входные данные: mov eax, <message>
47          <1> sprintf:
48 0000002D E8DDFFFF    <1> call    sprint
49
50 00000032 50          <1> push    eax
51 00000033 B80A000000   <1> mov     eax, 0Ah
52 00000038 50          <1> push    eax
53 00000039 89E0       <1> mov     eax, esp
54 0000003B E8CFFFFF    <1> call    sprintf
55 00000040 58         <1> pop     eax
56 00000041 58         <1> pop     eax
57 00000042 C3         <1> ret
58
59          <1> ;----- read -----

```

Рис. 3.5: Вывод файла листинга

6. Изменим код, как видим выводится ошибка, также в листинге в строке 24 она пишется.

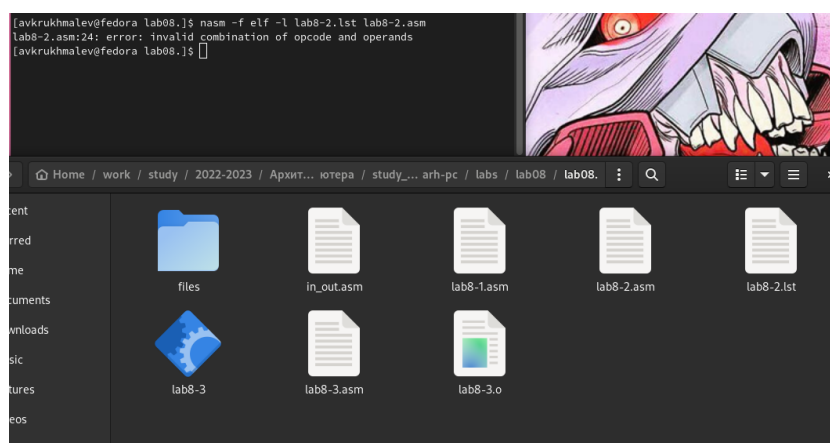


Рис. 3.6: Программа с ошибкой


```

23
24
24          *****      error: invalid combination of opcode and operands
25 00000110 890D[00000000]      mov [max],ecx
26
27 00000116 3B0D[39000000]      cmp ecx,[C]
28 0000011C 7F0C      jg check_B
29 0000011E 8B0D[39000000]      mov ecx,[C]
30 00000124 890D[00000000]      mov [max],ecx
31

```

Рис. 3.7: Её листинг

4 Самостоятельная работа

1. Мне попался 1 вариант, напишем код и выведем результат

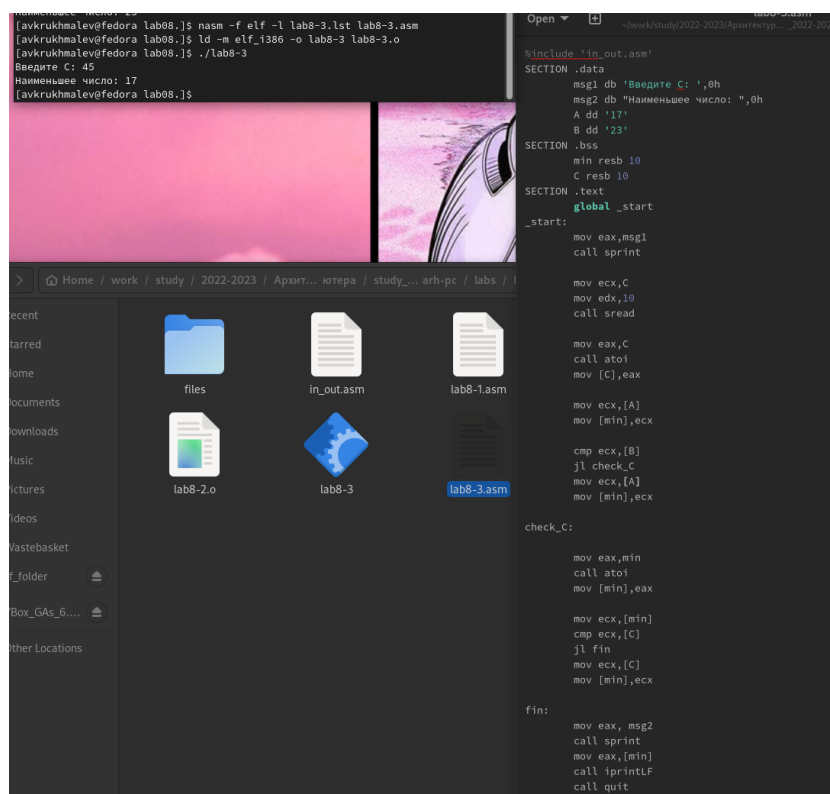


Рис. 4.1: 1-я задача

2. Сделаем 2-ю задачу

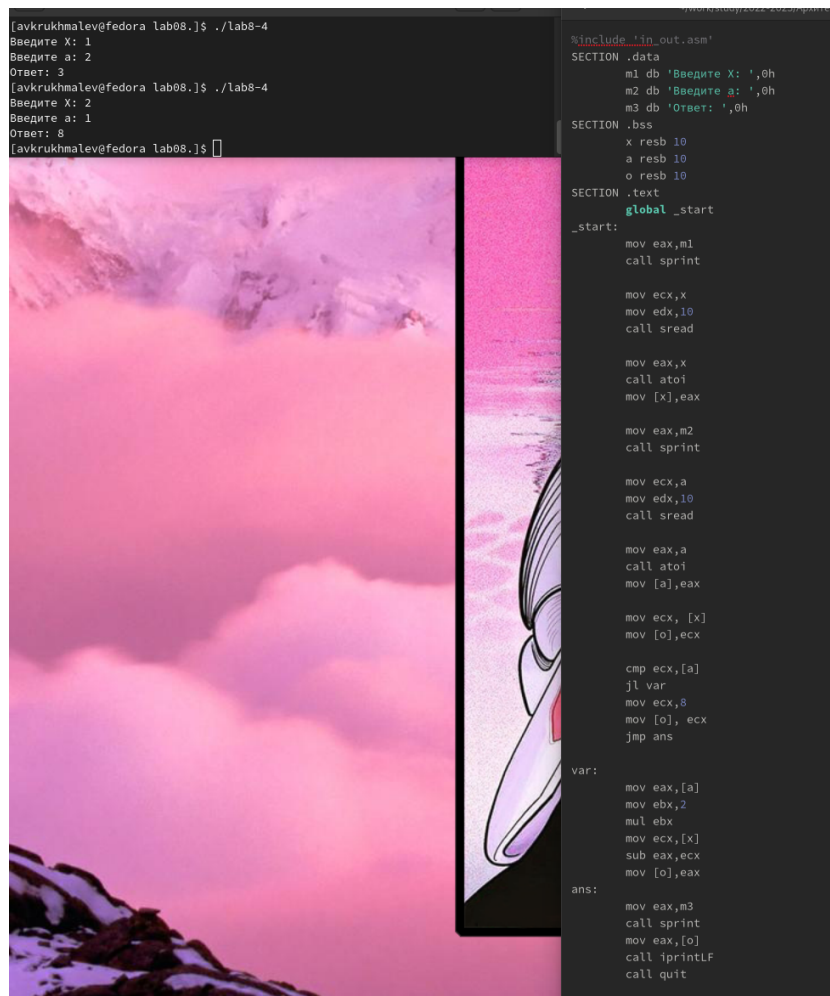


Рис. 4.2: 2-я задача

5 Выводы

В данной работе мы познакомились с безусловным и условным переходом, написали программы с использованием ветвления