

Отчёта по лабораторной работе 8

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

Кара-сал Эльдар Эдуардович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	22
	Список литературы	23

Список иллюстраций

2.1	Файл lab8-1.asm:	7
2.2	Программа lab8-1.asm:	8
2.3	Файл lab8-1.asm:	9
2.4	Программа lab8-1.asm:	10
2.5	Файл lab8-1.asm	11
2.6	Программа lab8-1.asm	12
2.7	Файл lab8-2.asm	13
2.8	Программа lab8-2.asm	14
2.9	Файл листинга lab8-2	15
2.10	ошибка трансляции lab8-2	16
2.11	файл листинга с ошибкой lab8-2	17
2.12	Файл lab8-3.asm	18
2.13	Программа lab8-3.asm	19
2.14	Файл lab8-4.asm	20
2.15	Программа lab8-4.asm	21

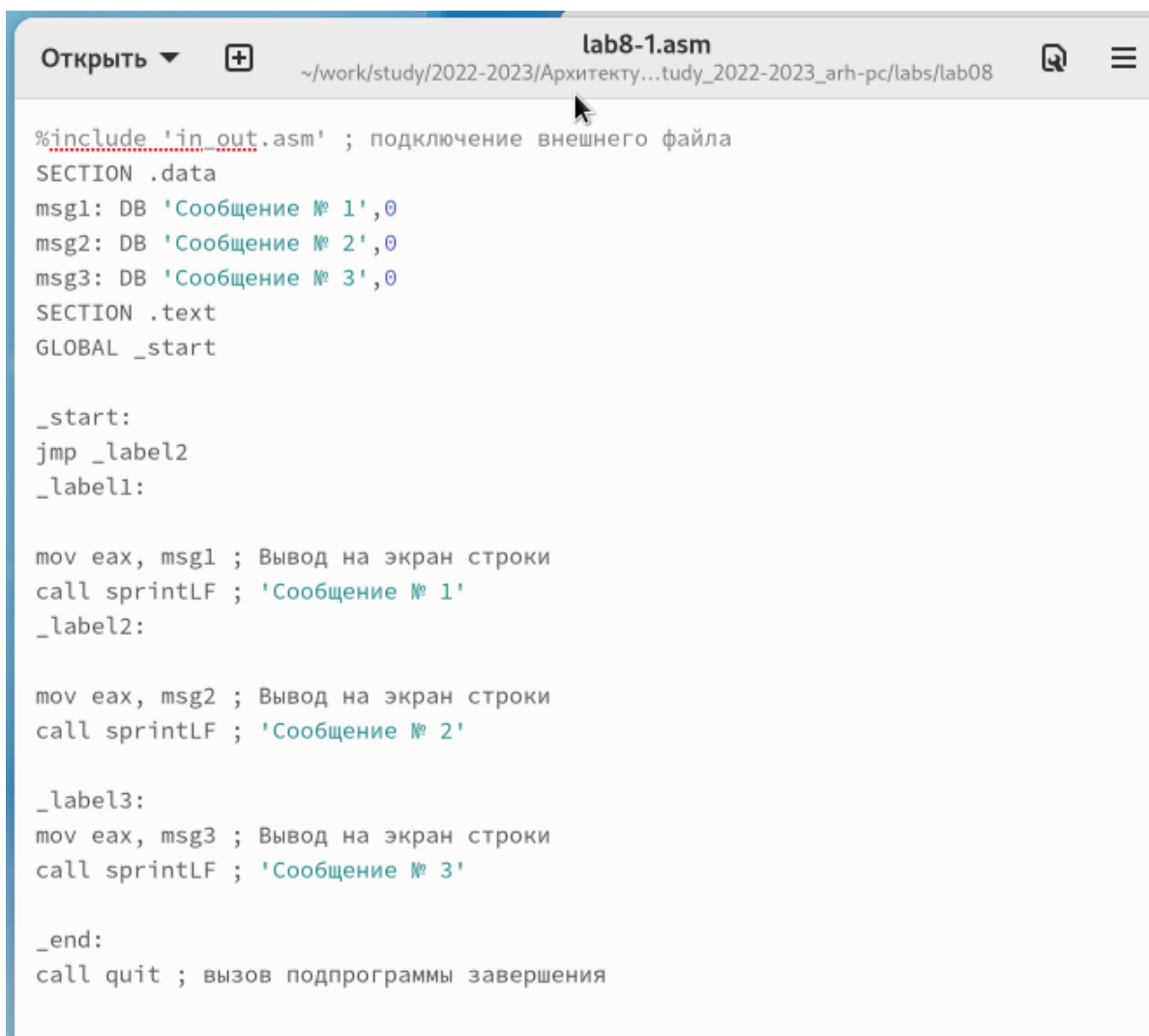
Список таблиц

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 № 8, перейдите в него и создайте файл lab8-1.asm
2. Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции jmp. Введите в файл lab8-1.asm текст программы из листинга 8.1. (рис. [2.1])



```
%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 sprintLF ; 'Сообщение № 1'
_label2:

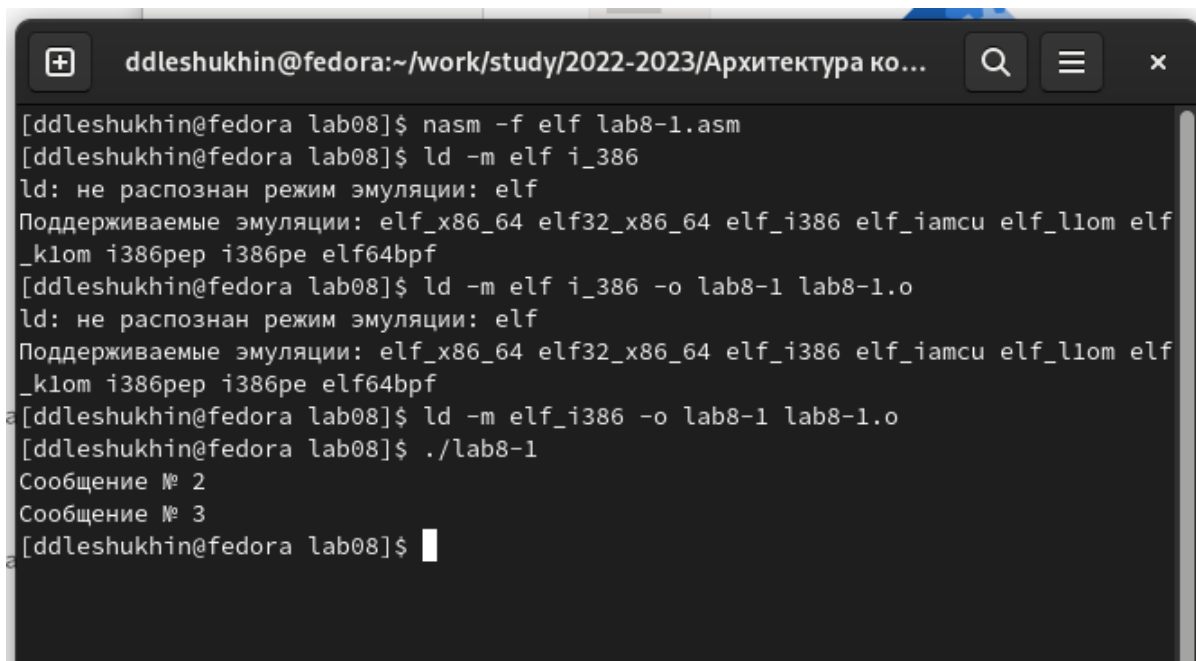
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'

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

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 2.1: Файл lab8-1.asm:

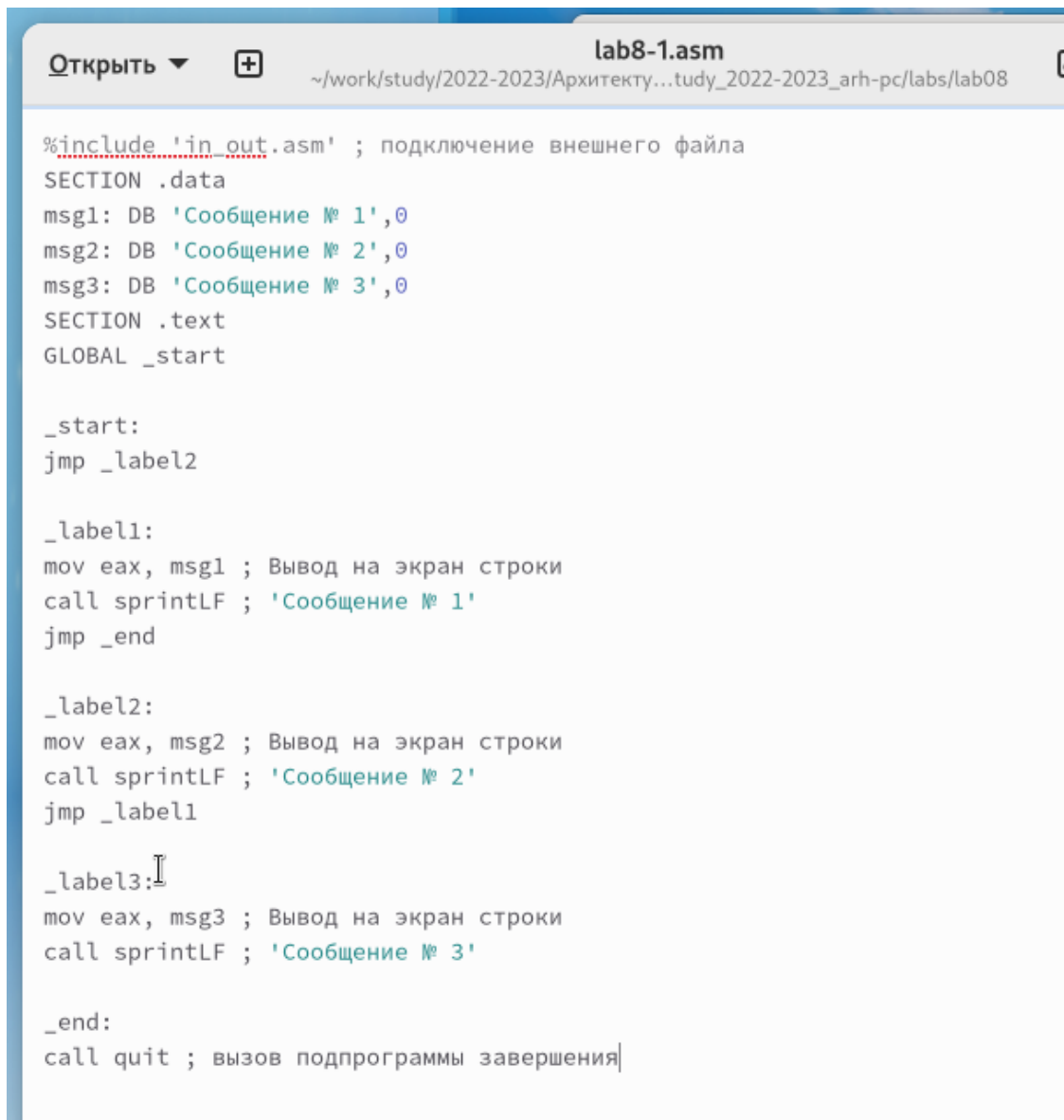
Создайте исполняемый файл и запустите его. (рис. [2.2])

A screenshot of a terminal window with a dark background. The window title is "ddleshukhin@fedora:~/work/study/2022-2023/Архитектура ко...". The terminal shows the following commands and output:

```
[ddleshukhin@fedora lab08]$ nasm -f elf lab8-1.asm
[ddleshukhin@fedora lab08]$ ld -m elf_i386
ld: не распознан режим эмуляции: elf
Поддерживаемые эмуляции: elf_x86_64 elf32_x86_64 elf_i386 elf_iamcu elf_llom elf_klom i386pep i386pe elf64bpf
[ddleshukhin@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
ld: не распознан режим эмуляции: elf
Поддерживаемые эмуляции: elf_x86_64 elf32_x86_64 elf_i386 elf_iamcu elf_llom elf_klom i386pep i386pe elf64bpf
[ddleshukhin@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[ddleshukhin@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[ddleshukhin@fedora lab08]$
```

Рис. 2.2: Программа lab8-1.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 8.2. (рис. [2.3], [2.4])



```
%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

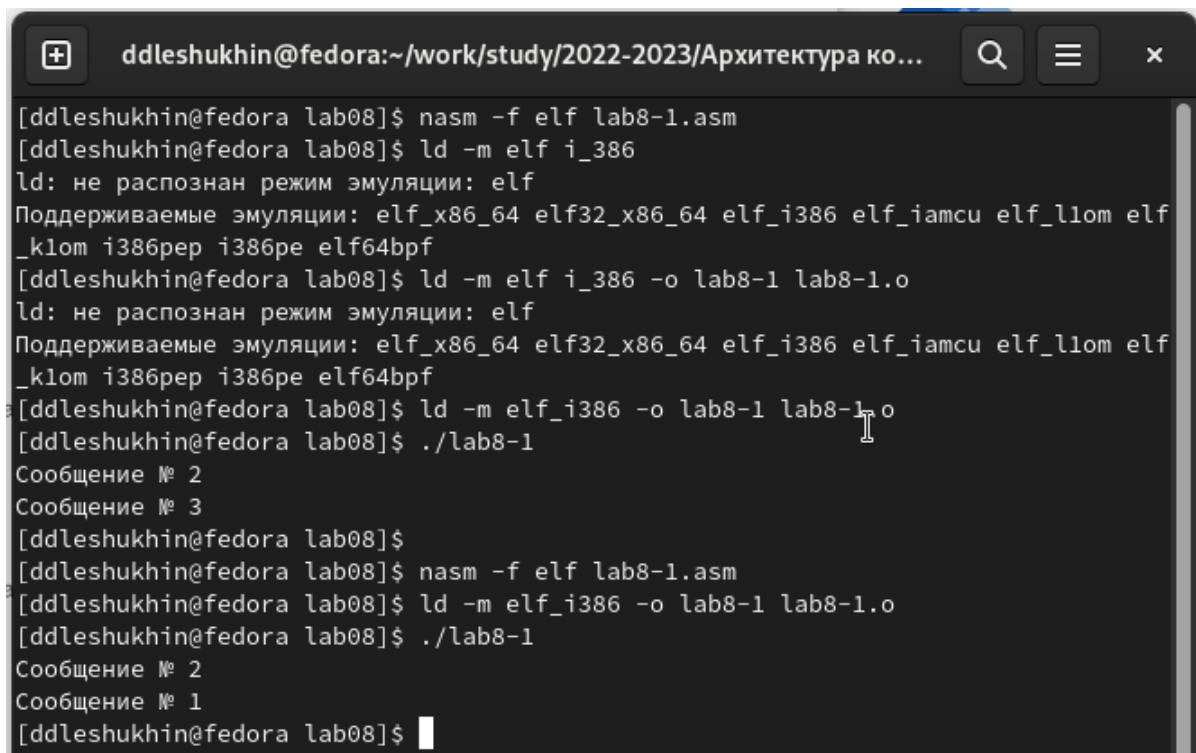
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end

_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 lab8-1.asm:

A terminal window with a dark background and light text. The window title is "ddleshukhin@fedora:~/work/study/2022-2023/Архитектура ко...". The terminal shows the following commands and output:

```
[ddleshukhin@fedora lab08]$ nasm -f elf lab8-1.asm
[ddleshukhin@fedora lab08]$ ld -m elf i_386
ld: не распознан режим эмуляции: elf
Поддерживаемые эмуляции: elf_x86_64 elf32_x86_64 elf_i386 elf_iamcu elf_llom elf
_klom i386pep i386pe elf64bpf
[ddleshukhin@fedora lab08]$ ld -m elf i_386 -o lab8-1 lab8-1.o
ld: не распознан режим эмуляции: elf
Поддерживаемые эмуляции: elf_x86_64 elf32_x86_64 elf_i386 elf_iamcu elf_llom elf
_klom i386pep i386pe elf64bpf
[ddleshukhin@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[ddleshukhin@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[ddleshukhin@fedora lab08]$
[ddleshukhin@fedora lab08]$ nasm -f elf lab8-1.asm
[ddleshukhin@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[ddleshukhin@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1
[ddleshukhin@fedora lab08]$
```

Рис. 2.4: Программа lab8-1.asm:

Измените текст программы добавив или изменив инструкции `jmp`, чтобы вывод программы был следующим (рис. [2.5], [2.6]):

Сообщение № 3

Сообщение № 2

Сообщение № 1

```
Открыть ▾ + lab8-1.asm ~/.work/study/2022-2023/Архитекту...tudy_2022-2023_arh-pc/labs/lab08

%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

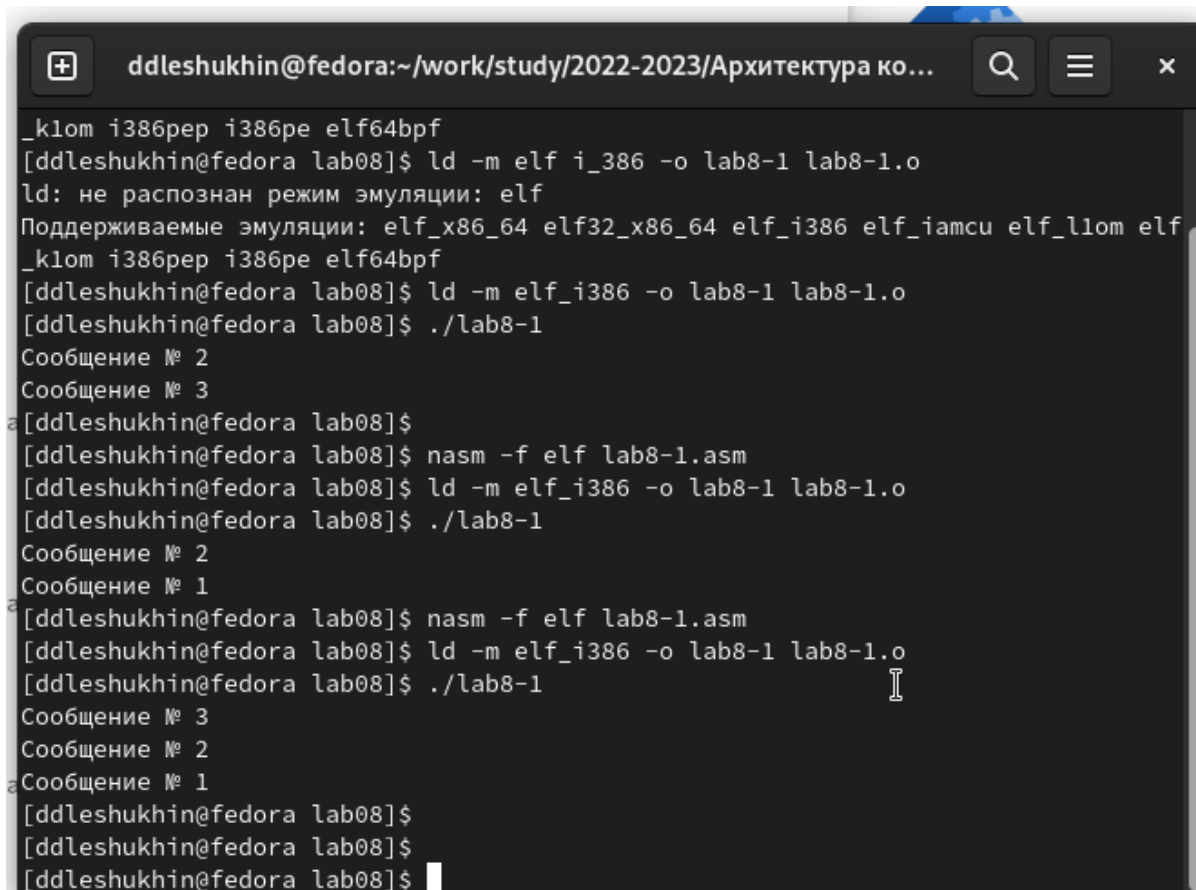
_label1:
mov eax, msg1 ; Вывод на экран строки
call printf ; 'Сообщение № 1'
jmp _end

_label2:
mov eax, msg2 ; Вывод на экран строки
call printf ; 'Сообщение № 2'
jmp _label1

_label3:
mov eax, msg3 ; Вывод на экран строки
call printf ; 'Сообщение № 3'
jmp _label2

_end:
call quit ; вызов подпрограммы завершения
```

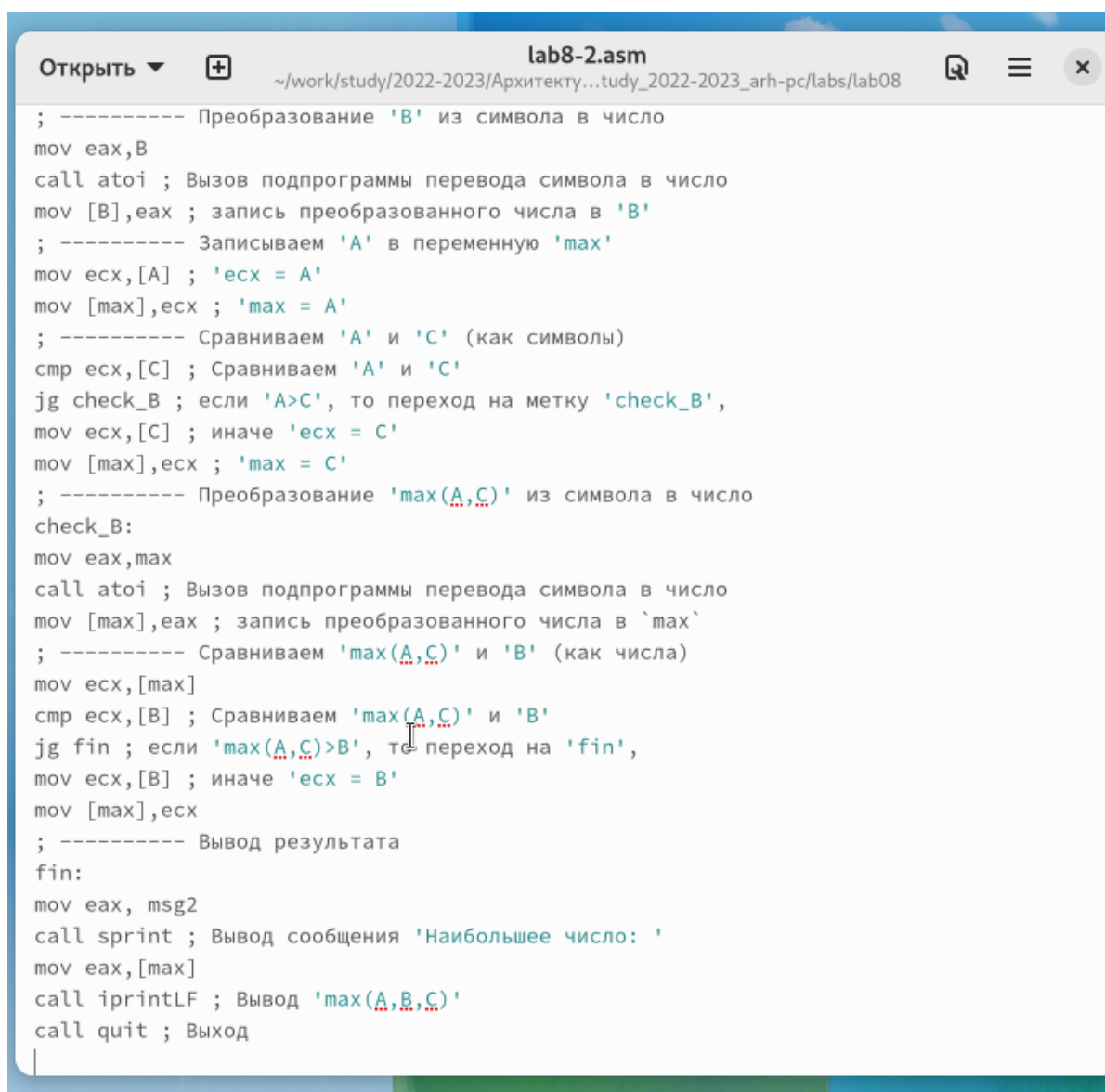
Рис. 2.5: Файл lab8-1.asm



```
ddleshukhin@fedora:~/work/study/2022-2023/Архитектура ко...
_klom i386pep i386pe elf64bpf
[ddleshukhin@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
ld: не распознан режим эмуляции: elf
Поддерживаемые эмуляции: elf_x86_64 elf32_x86_64 elf_i386 elf_iamcu elf_llom elf
_klom i386pep i386pe elf64bpf
[ddleshukhin@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[ddleshukhin@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[ddleshukhin@fedora lab08]$
[ddleshukhin@fedora lab08]$ nasm -f elf lab8-1.asm
[ddleshukhin@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[ddleshukhin@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1
[ddleshukhin@fedora lab08]$ nasm -f elf lab8-1.asm
[ddleshukhin@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[ddleshukhin@fedora lab08]$ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[ddleshukhin@fedora lab08]$
[ddleshukhin@fedora lab08]$
[ddleshukhin@fedora lab08]$
```

Рис. 2.6: Программа lab8-1.asm

- 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. (рис. [2.7], [2.8])



```
Открыть + lab8-2.asm
~/work/study/2022-2023/Архитекту...tudy_2022-2023_arh-pc/labs/lab08

; ----- 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 sprintf ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рис. 2.7: Файл lab8-2.asm

```
[ddleshukhin@fedora lab08]$  
[ddleshukhin@fedora lab08]$ nasm -f elf lab8-2.asm  
[ddleshukhin@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o  
[ddleshukhin@fedora lab08]$ ./lab8-2  
Введите B: 150  
Наибольшее число: 150  
[ddleshukhin@fedora lab08]$ ./lab8-2  
Введите B: 100  
Наибольшее число: 100  
[ddleshukhin@fedora lab08]$ ./lab8-2  
Введите B: 40  
Наибольшее число: 50  
[ddleshukhin@fedora lab08]$
```

Рис. 2.8: Программа lab8-2.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 `lab8-2.asm` (рис. [2.9])

```
40 0000002B 5A <1> pop edx
41 0000002C C3 <1> ret
42 <1>
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 E8DDFFFFFF <1> call sprint
49 <1>
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 E8CFFFFFFF <1> call sprint
55 00000040 58 <1> pop eax
56 00000041 58 <1> pop eax
57 00000042 C3 <1> ret
58 <1>
59 <1> ;----- sread -----
60 <1> ; Функция считывания сообщения
61 <1> ; входные данные: mov eax,<buffer>, mov ebx,<N>
62 <1> sread:
63 00000043 53 <1> push ebx
64 00000044 50 <1> push eax
65 <1>
66 00000045 BB00000000 <1> mov ebx, 0
67 0000004A B803000000 <1> mov eax, 3
68 0000004F CD80 <1> int 80h
```

Рис. 2.9: Файл листинга lab8-2

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

строка 51

- 51 - номер строки
- 00000033 - адрес
- B80A000000 - машинный код
- mov eax, 0AH - код программы

строка 52

- 52 - номер строки

- 00000038 - адрес
- 50 - машинный код
- push eax- код программы

строка 53

- 53 - номер строки
- 00000039 - адрес
- 89E0 - машинный код
- mov eax, esp - код программы

Откройте файл с программой lab8-2.asm и в любой инструкции с двумя операндами удалить один операнд. Выполните трансляцию с получением файла листинга (рис. [2.10],[2.11])

```
[ddleshukhin@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-2.lst
[ddleshukhin@fedora lab08]$
[ddleshukhin@fedora lab08]$
[ddleshukhin@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-2.lst
lab8-2.asm:34: error: invalid combination of opcode and operands
[ddleshukhin@fedora lab08]$
[ddleshukhin@fedora lab08]$
```

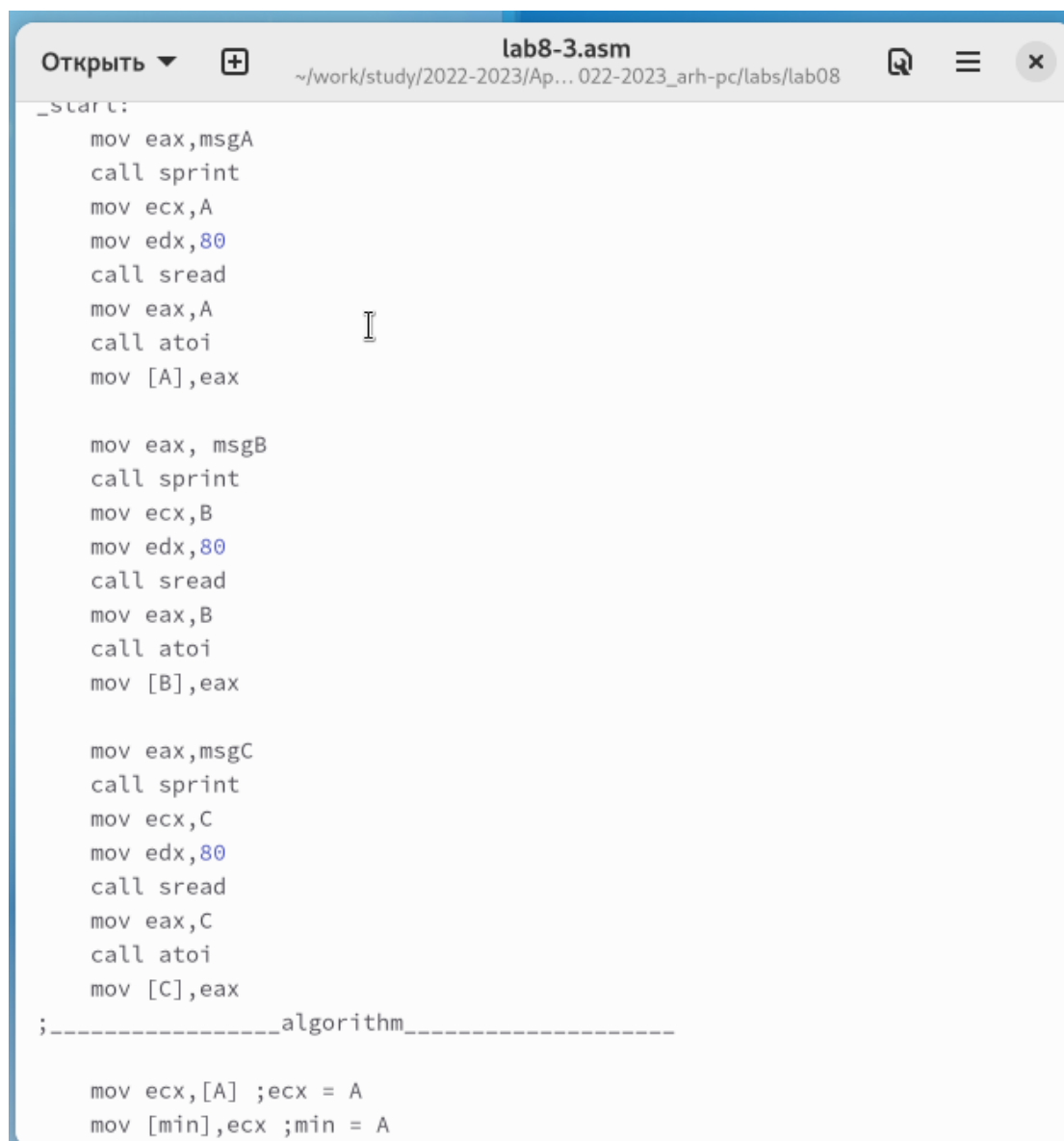
Рис. 2.10: ошибка трансляции lab8-2


```
lab8-2.asm                                lab8-2.lst
21 00000101 00[00000000]                          mov eax,0
22 00000106 E891FFFFFF                          call atoi ; Вызов подпрограммы перевода символа в число
23 0000010B A3[0A000000]                          mov [B],eax ; запись преобразованного числа в 'B'
24                                     ; ----- Записываем 'A' в переменную 'max'
25 00000110 8B0D[35000000]                          mov ecx,[A] ; 'ecx' = A
26 00000116 890D[00000000]                          mov [max],ecx ; 'max' = A
27                                     ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000]                          cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 00000122 7F0C                               jg check_B ; если 'A'>'C', то переход на метку 'check_B',
30 00000124 8B0D[39000000]                          mov ecx,[C] ; иначе 'ecx' = C
31 0000012A 890D[00000000]                          mov [max],ecx ; 'max' = C
32                                     ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B                               mov eax,
34                                     error: invalid combination of opcode and operands
35 00000130 E867FFFFFF                          call atoi ; Вызов подпрограммы перевода символа в число
36 00000135 A3[00000000]                          mov [max],eax ; запись преобразованного числа в 'max'
37                                     ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013A 8B0D[00000000]                          mov ecx,[max]
39 00000140 3B0D[0A000000]                          cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 00000146 7F0C                               jg fin ; если 'max(A,C)>'B', то переход на 'fin',
41 00000148 8B0D[0A000000]                          mov ecx,[B] ; иначе 'ecx' = B
42 0000014E 890D[00000000]                          mov [max],ecx
43                                     ; ----- Вывод результата
44 fin:
45 00000154 B8[13000000]                          mov eax, msg2
46 00000159 E8B1FFFFFF                          call sprintf ; Вывод сообщения 'Наибольшее число: '
47 0000015E A1[00000000]                          mov eax,[max]
48 00000163 E81EFFFFFF                          call iprintLF ; Вывод 'max(A,B,C)'
```

Рис. 2.11: файл листинга с ошибкой lab8-2

5. Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и c. Значения переменных выбрать из табл. 8.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу (рис. [2.12],[2.13])

для варианта 10 - 41, 62, 35



```
Открыть ▾ + lab8-3.asm
~/work/study/2022-2023/Ар... 022-2023_arh-pc/labs/lab08

_start:
    mov eax,msgA
    call sprint
    mov ecx,A
    mov edx,80
    call sread
    mov eax,A
    call atoi
    mov [A],eax

    mov eax, msgB
    call sprint
    mov ecx,B
    mov edx,80
    call sread
    mov eax,B
    call atoi
    mov [B],eax

    mov eax,msgC
    call sprint
    mov ecx,C
    mov edx,80
    call sread
    mov eax,C
    call atoi
    mov [C],eax

;-----algorithm-----

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

Рис. 2.12: Файл lab8-3.asm

```
[ddleshukhin@fedora lab08]$  
[ddleshukhin@fedora lab08]$ nasm -f elf lab8-3.asm  
[ddleshukhin@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o  
[ddleshukhin@fedora lab08]$ ./lab8-3  
Input A: 81  
Input B: 22  
Input C: 72  
Smallest: 22  
[ddleshukhin@fedora lab08]$  
[ddleshukhin@fedora lab08]$
```

Рис. 2.13: Программа lab8-3.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 8.6 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 8.6. (рис. [2.14],[2.15])

для варианта 10

$$\begin{cases} x - 2, x > 2 \\ 3a, x < 2 \end{cases}$$

```
lab8-4.asm
~/work/study/2022-2023/Ap... 022-2023_arh-pc/labs/lab08

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,3
mul ebx
add eax,1
call iprintLF
call quit

second:
mov eax,[X]
mov ebx,3
mul ebx
add eax,1
call iprintLF
call quit
```

Рис. 2.14: Файл lab8-4.asm

```
[ddleshukhin@fedora lab08]$  
[ddleshukhin@fedora lab08]$  
[ddleshukhin@fedora lab08]$ nasm -f elf lab8-4.asm  
[ddleshukhin@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o  
[ddleshukhin@fedora lab08]$ ./lab8-4  
Input A: 3  
Input X: 2  
10  
[ddleshukhin@fedora lab08]$ ./lab8-4  
Input A: 2  
Input X: 4  
13  
[ddleshukhin@fedora lab08]$
```

Рис. 2.15: Программа lab8-4.asm

3 Выводы

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

Список литературы

1. Расширенный ассемблер: NASM
2. MASM, TASM, FASM, NASM под Windows и Linux