

Отчёта по лабораторной работе 8

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

Аристил Линдсэй Виллиам

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	20

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

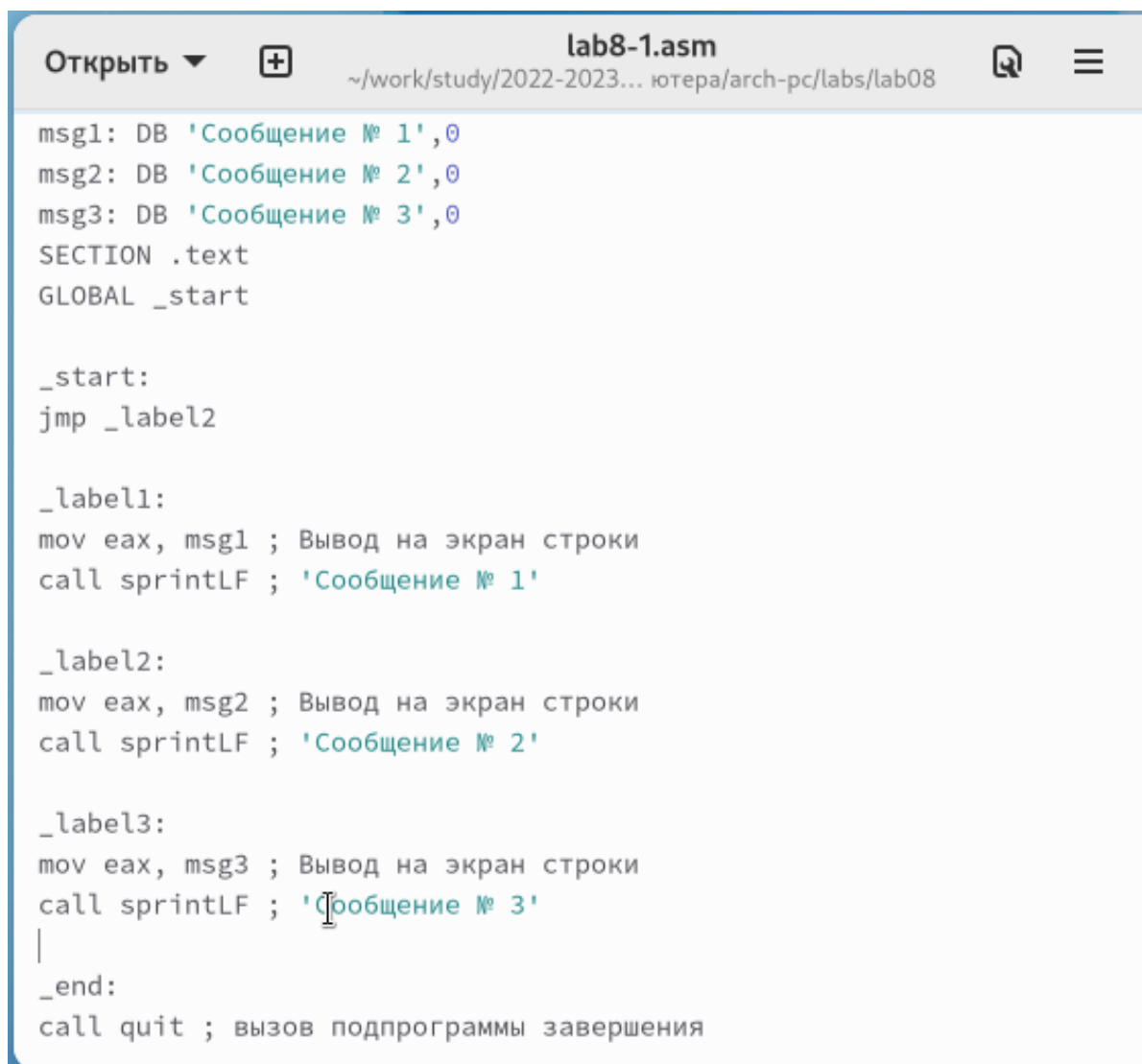
Список таблиц

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)



```
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start

_start:
jmp _label2

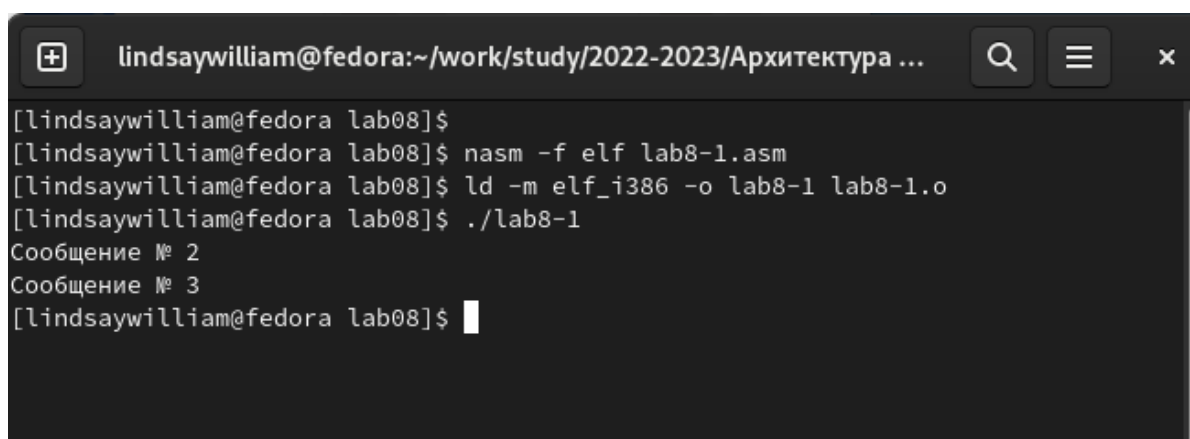
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 1'

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

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 3'
|
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 2.1: Файл lab8-1.asm:

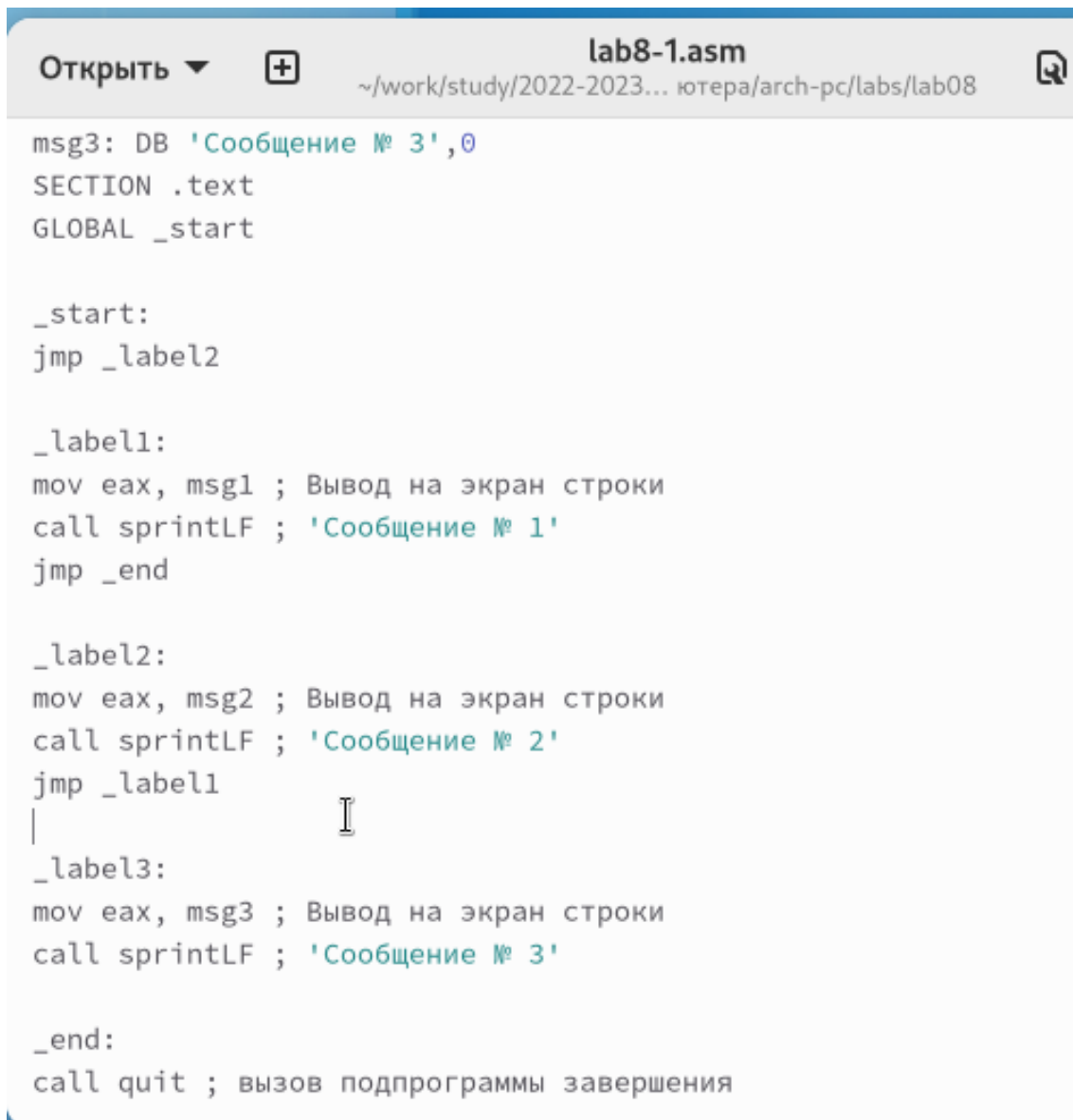
Создайте исполняемый файл и запустите его. (рис. 2.2)

A terminal window with a dark background. The title bar shows the user 'lindsaywilliam@fedora' and the current directory '~/work/study/2022-2023/Архитектура ...'. The terminal contains the following text:

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



```
lab8-1.asm
~/work/study/2022-2023... ютеpa/arch-пс/labs/lab08

msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start

_start:
jmp _label2

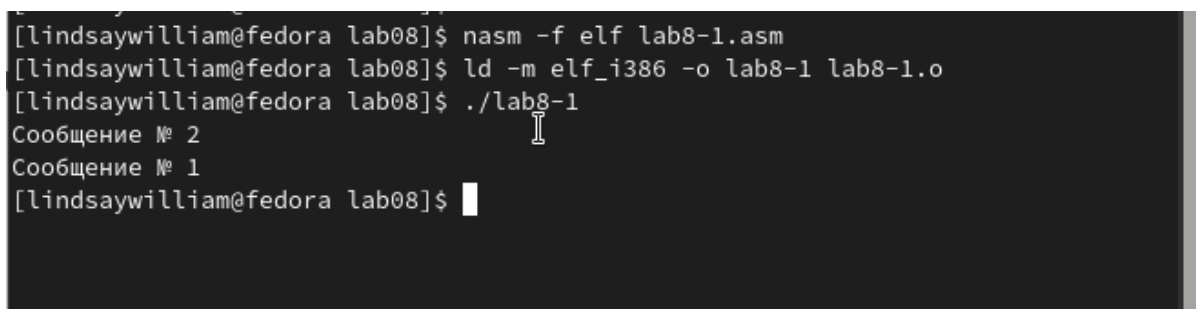
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end

_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:



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

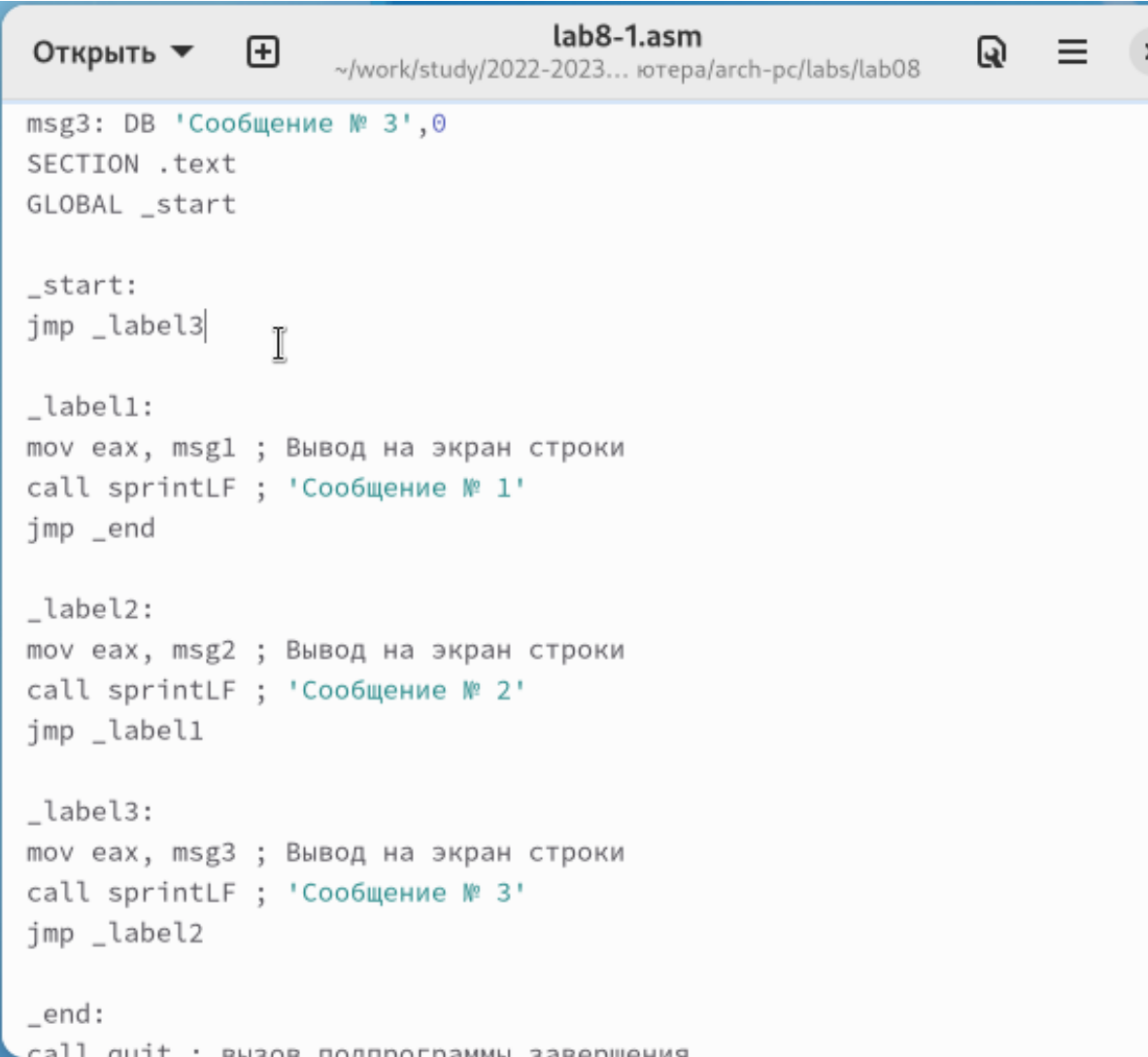
Рис. 2.4: Программа lab8-1.asm:

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

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
lab8-1.asm
~/work/study/2022-2023... ютера/arch-pc/labs/lab08

msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start

_start:
jmp _label3

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

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

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

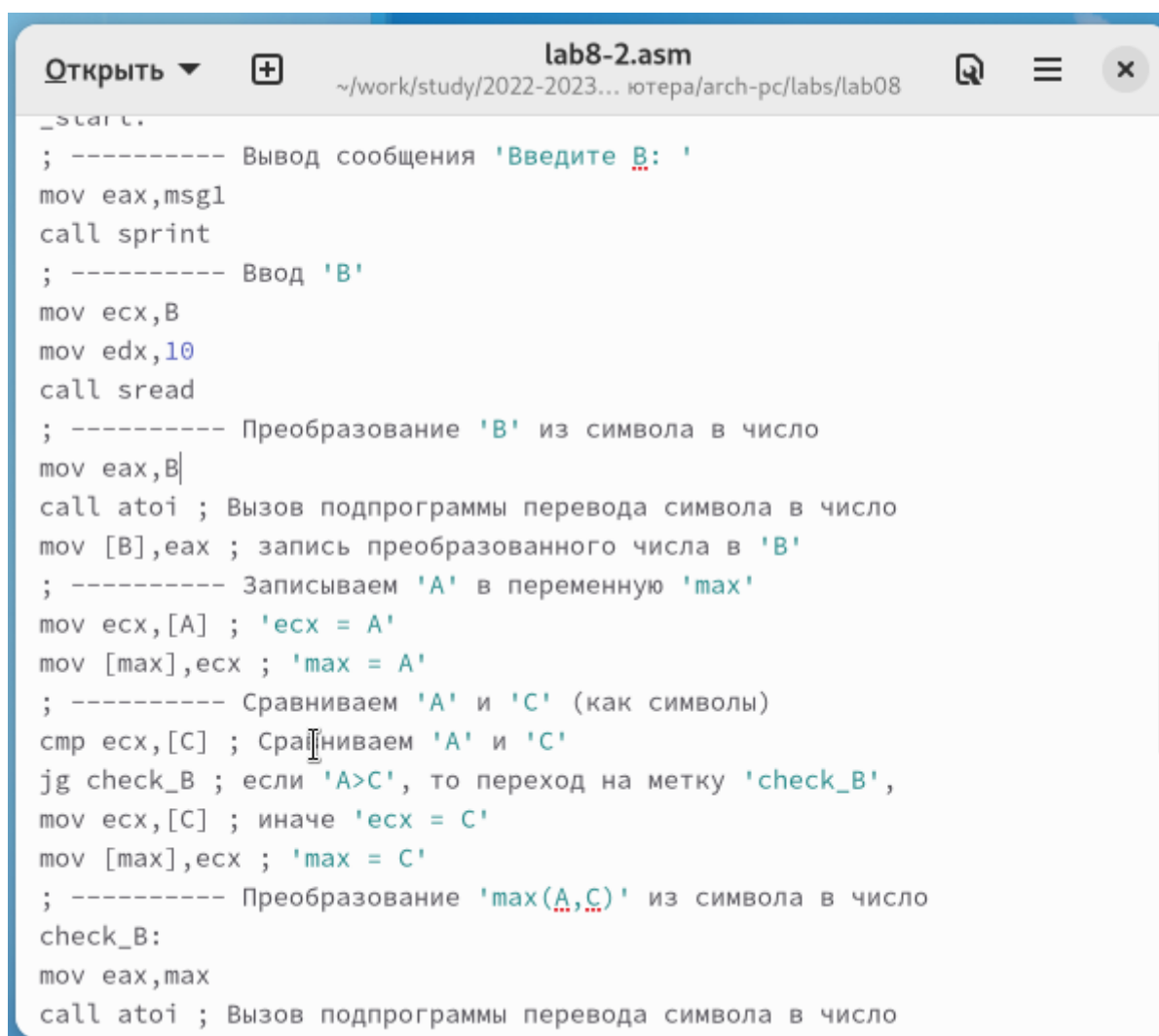
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 2.5: Файл lab8-1.asm

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

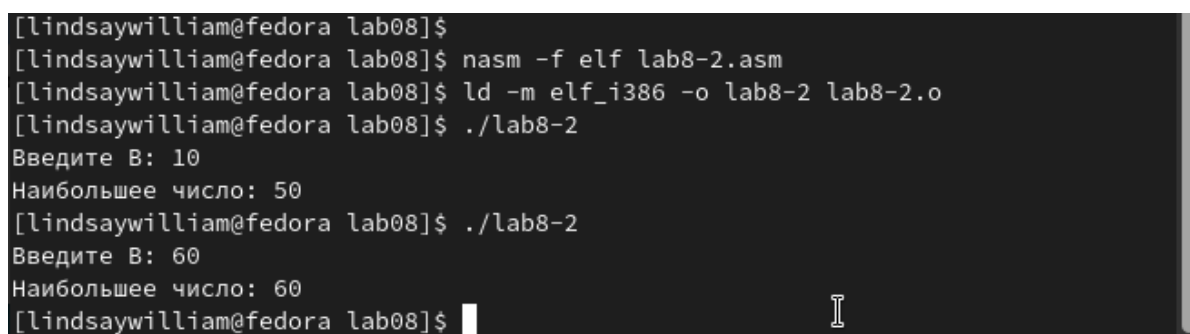
Рис. 2.6: Программа lab8-1.asm

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. (рис. 2.7, 2.8)



```
Открыть + lab8-2.asm ~/work/study/2022-2023... ютера/arch-pc/labs/lab08
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
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
```

Рис. 2.7: Файл lab8-2.asm




```
[lindsaywilliam@fedora lab08]$
[lindsaywilliam@fedora lab08]$ nasm -f elf lab8-2.asm
[lindsaywilliam@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[lindsaywilliam@fedora lab08]$ ./lab8-2
Введите B: 10
Наибольшее число: 50
[lindsaywilliam@fedora lab08]$ ./lab8-2
Введите B: 60
Наибольшее число: 60
[lindsaywilliam@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)



```
lab8-2.lst
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08

lab8-2.asm
lab8-2.lst

6 00000039 35300000      C dd '50'
7                          section .bss
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 000000F2 B9[0A000000]      mov ecx, B
18 000000F7 BA0A000000      mov edx, 10
19 000000FC E842FFFFFF      call sread
20                          ; ----- Преобразование 'B' из символа в число
21 00000101 B8[0A000000]      mov eax, B
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:
34 00000130 B8[00000000]      mov eax, max
35 00000135 E862FFFFFF      call atoi ; Вызов подпрограммы перевода символа в число
36 0000013A A3[00000000]      mov [max], eax ; запись преобразованного числа в 'max'
```

Рис. 2.9: Файл листинга lab8-2

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

строка 10

- 10 - номер строки
- 00000006 - адрес
- 7403 - машинный код
- jz finished - код программы

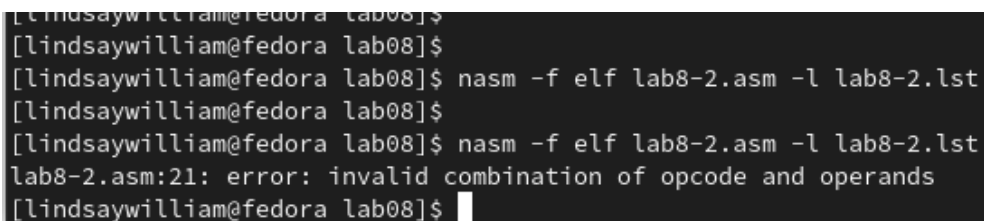
строка 11

- 11 - номер строки
- 00000008 - адрес
- 40 - машинный код
- inc eax - код программы

строка 12

- 12 - номер строки
- 00000009 - адрес
- EBF8 - машинный код
- jmp nextchar - код программы

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



```
[lindsaywilliam@fedora lab08]$
[lindsaywilliam@fedora lab08]$
[lindsaywilliam@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-2.lst
[lindsaywilliam@fedora lab08]$
[lindsaywilliam@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-2.lst
lab8-2.asm:21: error: invalid combination of opcode and operands
[lindsaywilliam@fedora lab08]$
```

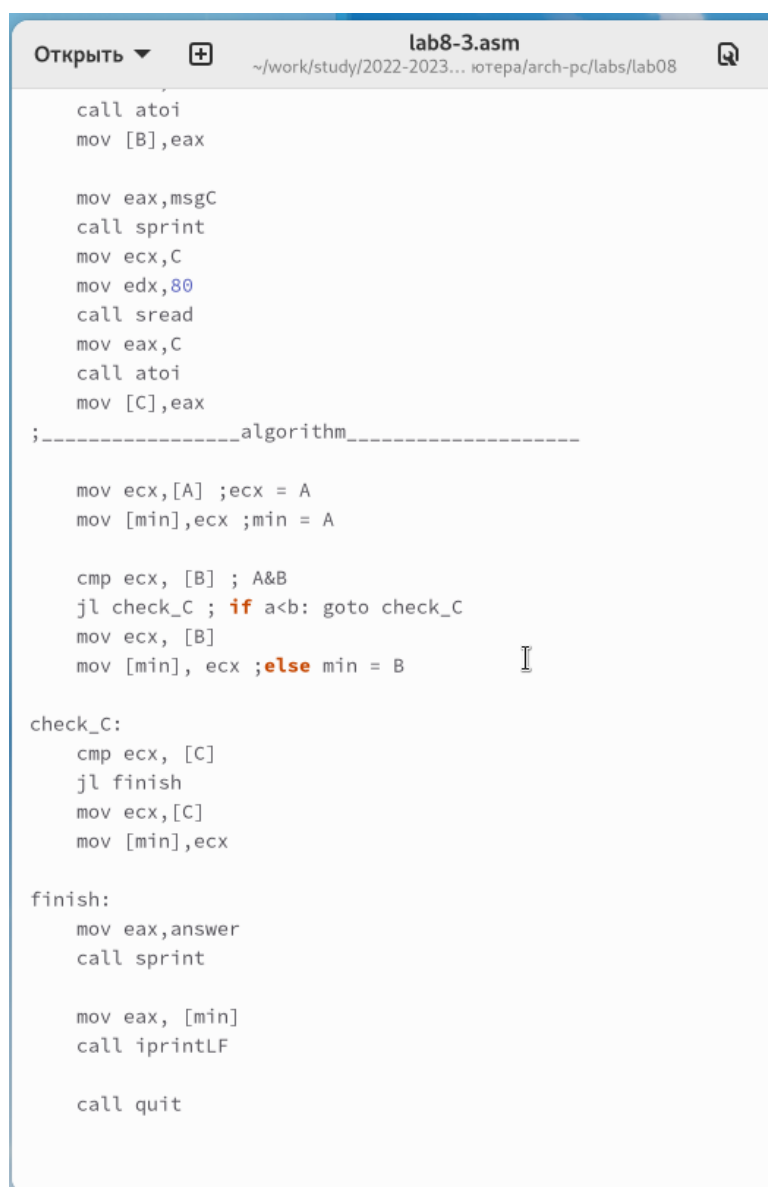
Рис. 2.10: ошибка трансляции lab8-2

```
15 000000ED E81DFFFFFF call sprint
16 ; ----- Ввод 'B'
17 000000F2 B90A000000 mov ecx, B
18 000000F7 BA0A000000 mov edx, 10
19 000000FC E842FFFFFF call sread
20 ; ----- Преобразование 'B' из символа в число
21 mov eax,
error: invalid combination of opcode and operands
22 00000101 E896FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
23 00000106 A30A000000 mov [B], eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'
25 0000010B 8B0D35000000 mov ecx, [A] ; 'ecx' = A
26 00000111 890D00000000 mov [max], ecx ; 'max' = A
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 00000117 3B0D39000000 cmp ecx, [C] ; Сравниваем 'A' и 'C'
29 0000011D 7F0C jg check_B ; если 'A>C', то переход на метку 'check_B'
30 0000011F 8B0D39000000 mov ecx, [C] ; иначе 'ecx' = C
31 00000125 890D00000000 mov [max], ecx ; 'max' = C
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 0000012B 8B0D00000000 mov eax, max
35 00000130 E867FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
36 00000135 A300000000 mov [max], eax ; запись преобразованного числа в 'max'
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013A 8B0D00000000 mov ecx, [max]
39 00000140 3B0D0A000000 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 8B0D0A000000 mov ecx, [B] ; иначе 'ecx' = B
42 0000014E 890D00000000 mov [max], ecx
43 ; ----- Вывод результата
```

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 12 - 99, 29, 26



```
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

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

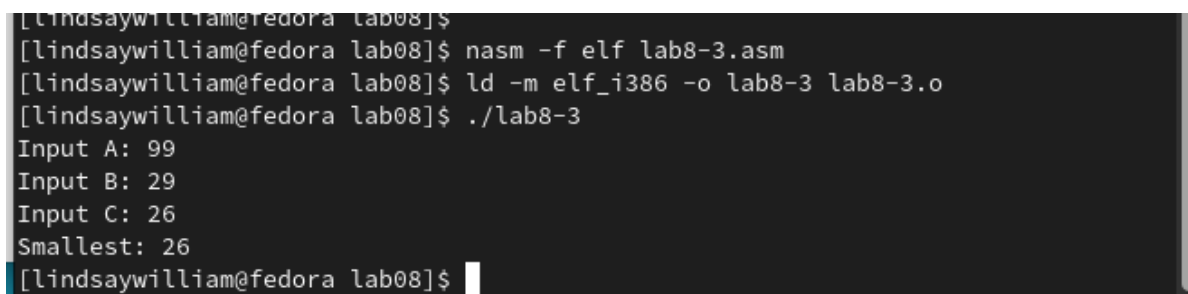
check_C:
cmp ecx, [C]
jl finish
mov ecx,[C]
mov [min],ecx

finish:
mov eax,answer
call sprint

mov eax, [min]
call iprintLF

call quit
```

Рис. 2.12: Файл lab8-3.asm



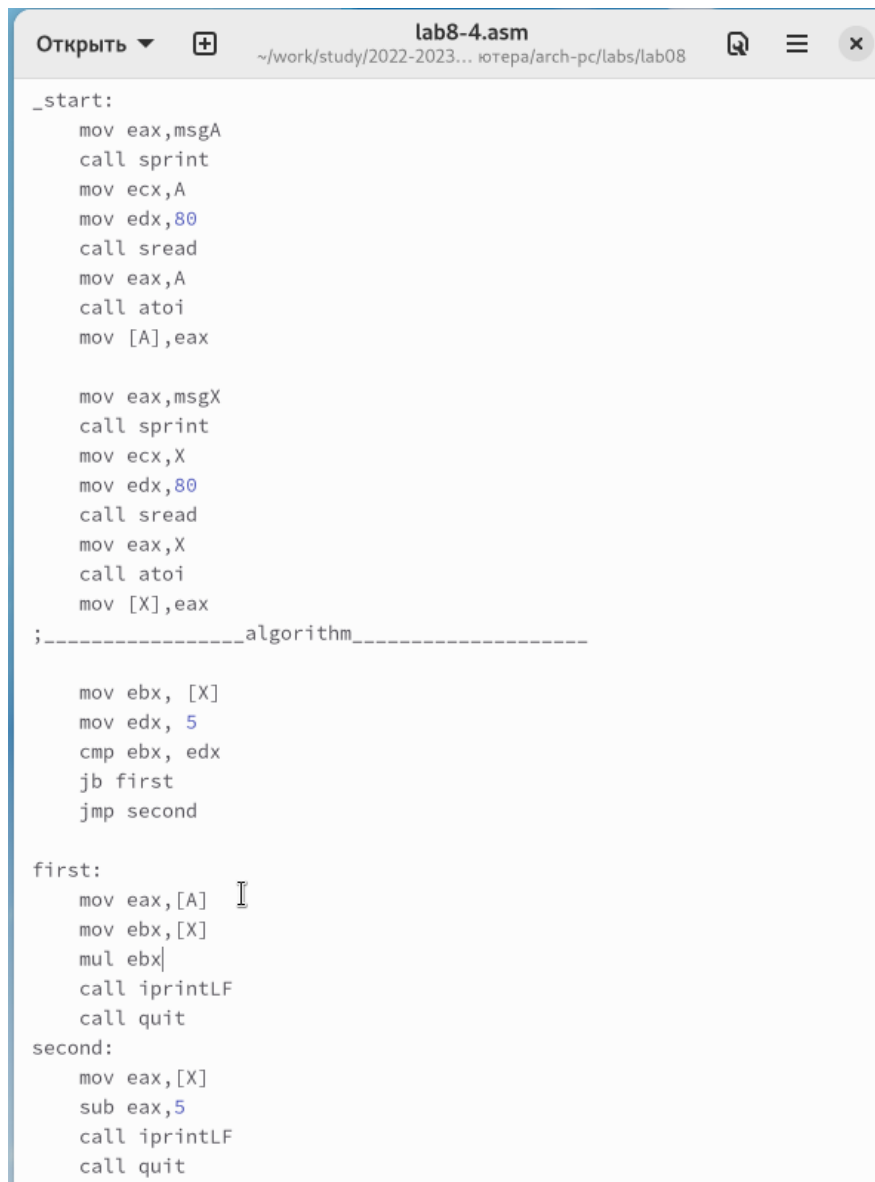
```
[lindsaywilliam@fedora lab08]$
[lindsaywilliam@fedora lab08]$ nasm -f elf lab8-3.asm
[lindsaywilliam@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[lindsaywilliam@fedora lab08]$ ./lab8-3
Input A: 99
Input B: 29
Input C: 26
Smallest: 26
[lindsaywilliam@fedora lab08]$
```

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 12

$$\begin{cases} a * x, x < 5 \\ x - 5, x \geq 5 \end{cases}$$



```
lab8-4.asm
~/work/study/2022-2023... ютеpa/arch-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,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, 5
    cmp ebx, edx
    jb first
    jmp second

first:
    mov eax,[A]
    mov ebx,[X]
    mul ebx
    call iprintLF
    call quit

second:
    mov eax,[X]
    sub eax,5
    call iprintLF
    call quit
```

Рис. 2.14: Файл lab8-4.asm

```
[lindsaywilliam@fedora lab08]$  
[lindsaywilliam@fedora lab08]$ nasm -f elf lab8-4.asm  
[lindsaywilliam@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o  
[lindsaywilliam@fedora lab08]$ ./lab8-4  
Input A: 7  
Input X: 3  
21  
[lindsaywilliam@fedora lab08]$ ./lab8-4  
Input A: 4  
Input X: 6  
1  
[lindsaywilliam@fedora lab08]$
```

Рис. 2.15: Программа lab8-4.asm

3 Выводы

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