

Отчёта по лабораторной работе 8

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

Туем Гислен НКАбд-03-22

Содержание

| | | |
|----------|---------------------------------------|-----------|
| 1 | Цель работы | 5 |
| 2 | Задание | 6 |
| 3 | Теоретическое введение | 7 |
| 4 | Выполнение лабораторной работы | 8 |
| 5 | Выводы | 24 |
| | Список литературы | 25 |

Список иллюстраций

| | | |
|------|--|----|
| 4.1 | Файл lab8-1.asm: | 9 |
| 4.2 | Программа lab8-1.asm: | 10 |
| 4.3 | Файл lab8-1.asm: | 11 |
| 4.4 | Программа lab8-1.asm: | 12 |
| 4.5 | Файл lab8-1.asm | 13 |
| 4.6 | Программа lab8-1.asm | 14 |
| 4.7 | Файл lab8-2.asm | 15 |
| 4.8 | Программа lab8-2.asm | 16 |
| 4.9 | Файл листинга lab8-2 | 17 |
| 4.10 | ошибка трансляции lab8-2 | 18 |
| 4.11 | файл листинга с ошибкой lab8-2 | 19 |
| 4.12 | Файл lab8-3.asm | 20 |
| 4.13 | Программа lab8-3.asm | 21 |
| 4.14 | Файл lab8-4.asm | 22 |
| 4.15 | Программа lab8-4.asm | 23 |

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 Задание

1. Изучите примеры программ.
2. Изучите файл листинга.
3. 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
4. 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.

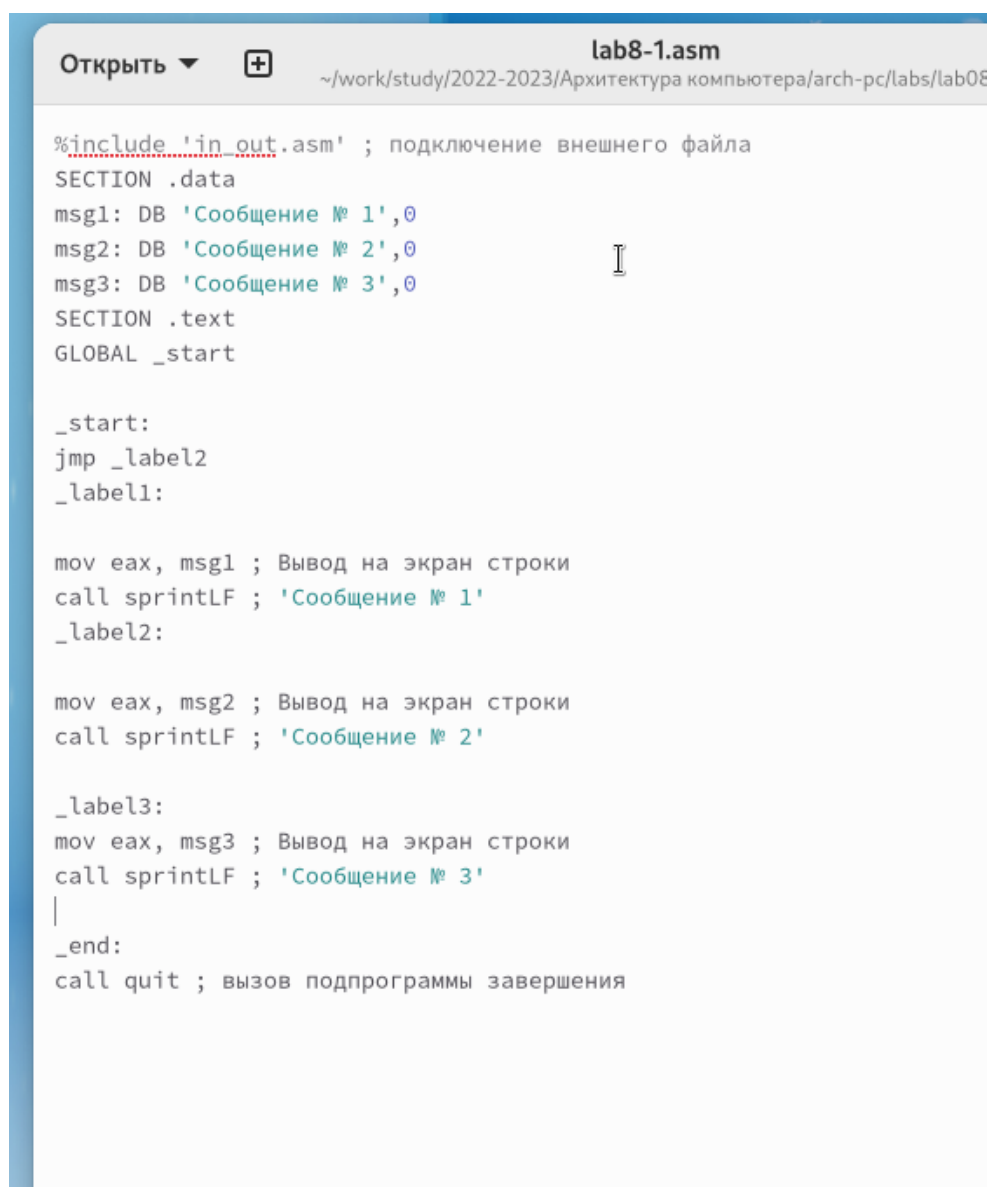
3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
- безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

4 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. 4.1)



```
Открыть ▾ + lab8-1.asm
~/work/study/2022-2023/Архитектура компьютера/arch-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 _label2
_label1:

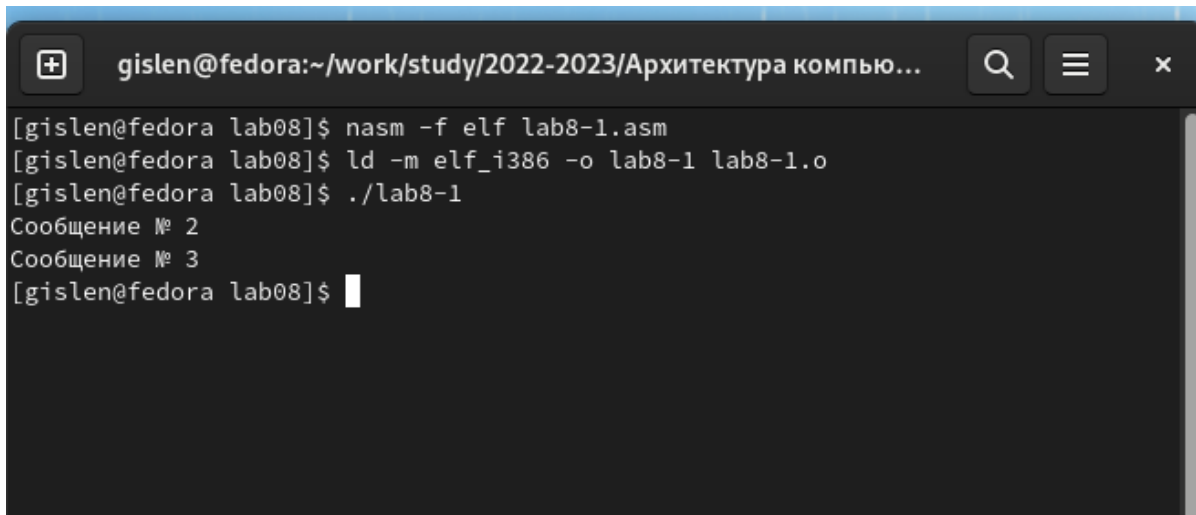
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
_label2:

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

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

Рис. 4.1: Файл lab8-1.asm:

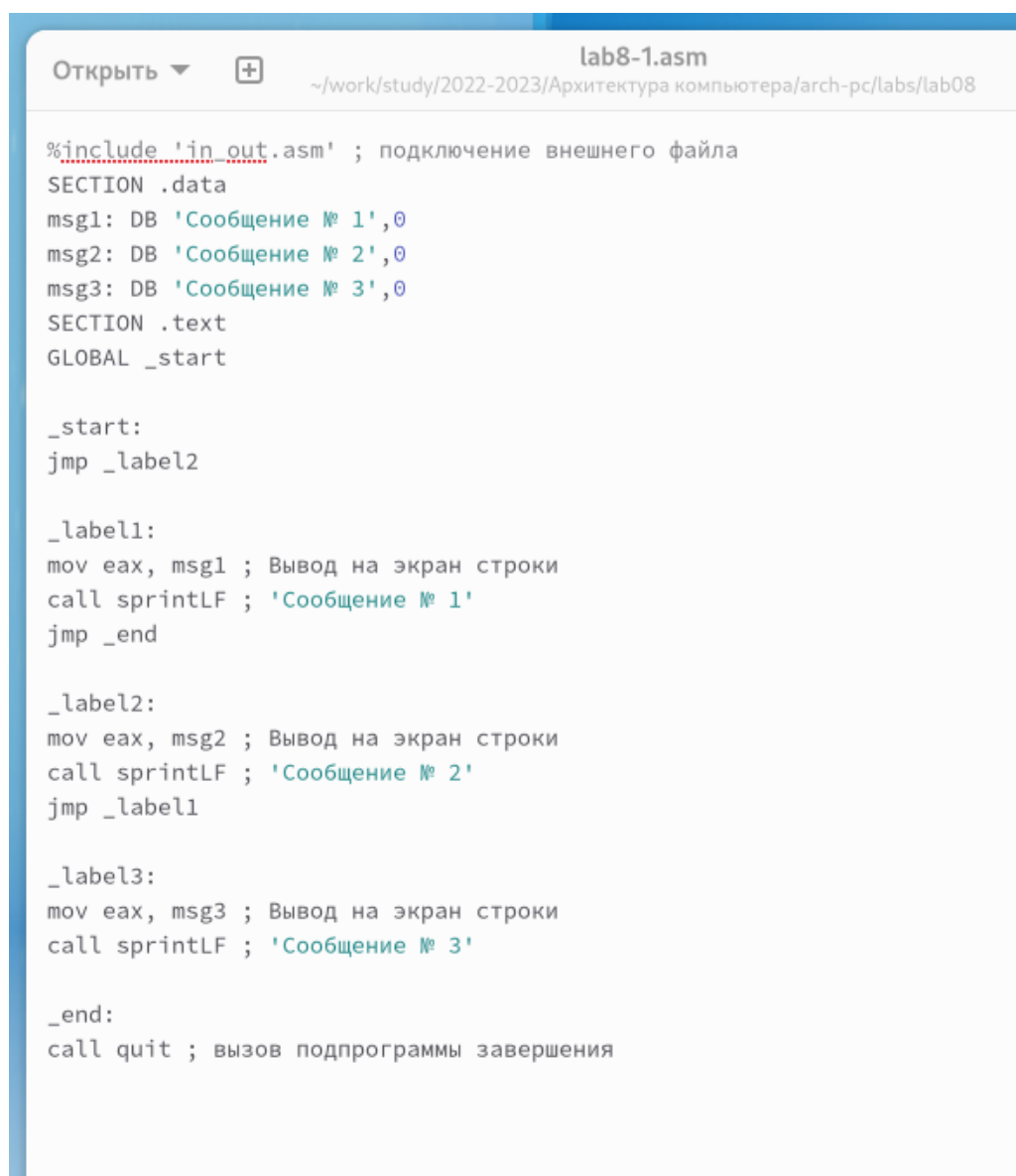
Создайте исполняемый файл и запустите его. (рис. 4.2)

A terminal window with a dark background and light text. The title bar shows the user 'gislen' on a 'fedora' machine, in the directory '~/work/study/2022-2023/Архитектура компью...'. The terminal contains the following text:

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

Рис. 4.2: Программа lab8-1.asm:

Здесь 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. 4.3, 4.4)



```
%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

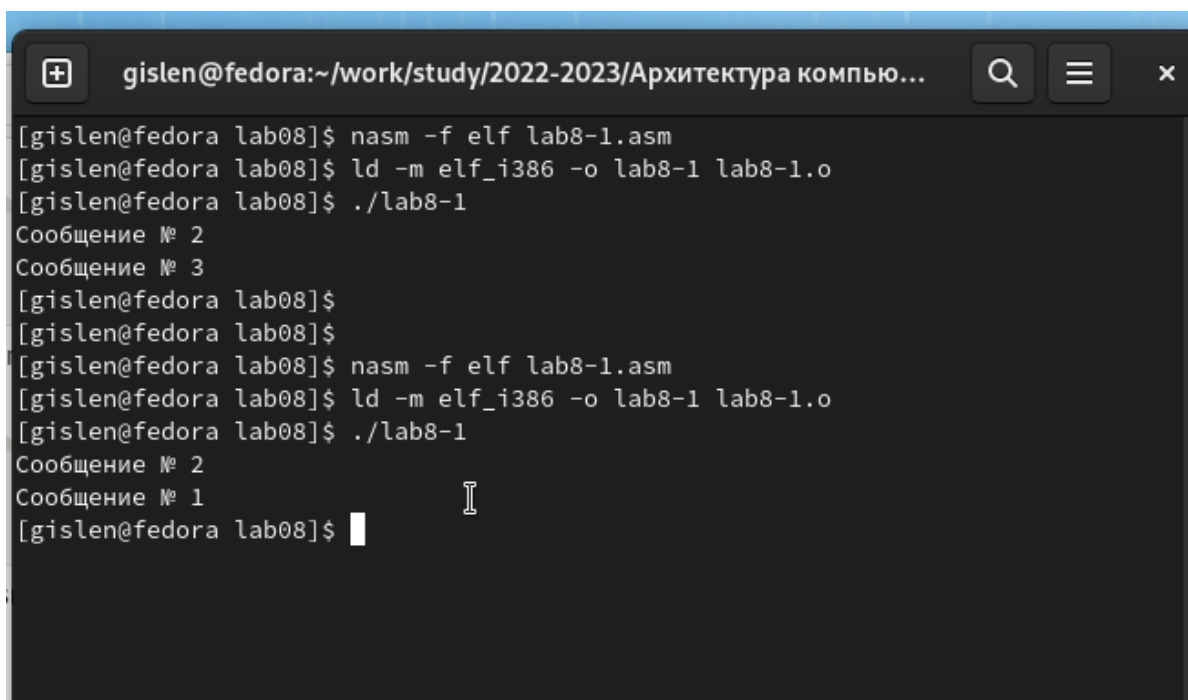
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end

_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. 4.3: Файл lab8-1.asm:

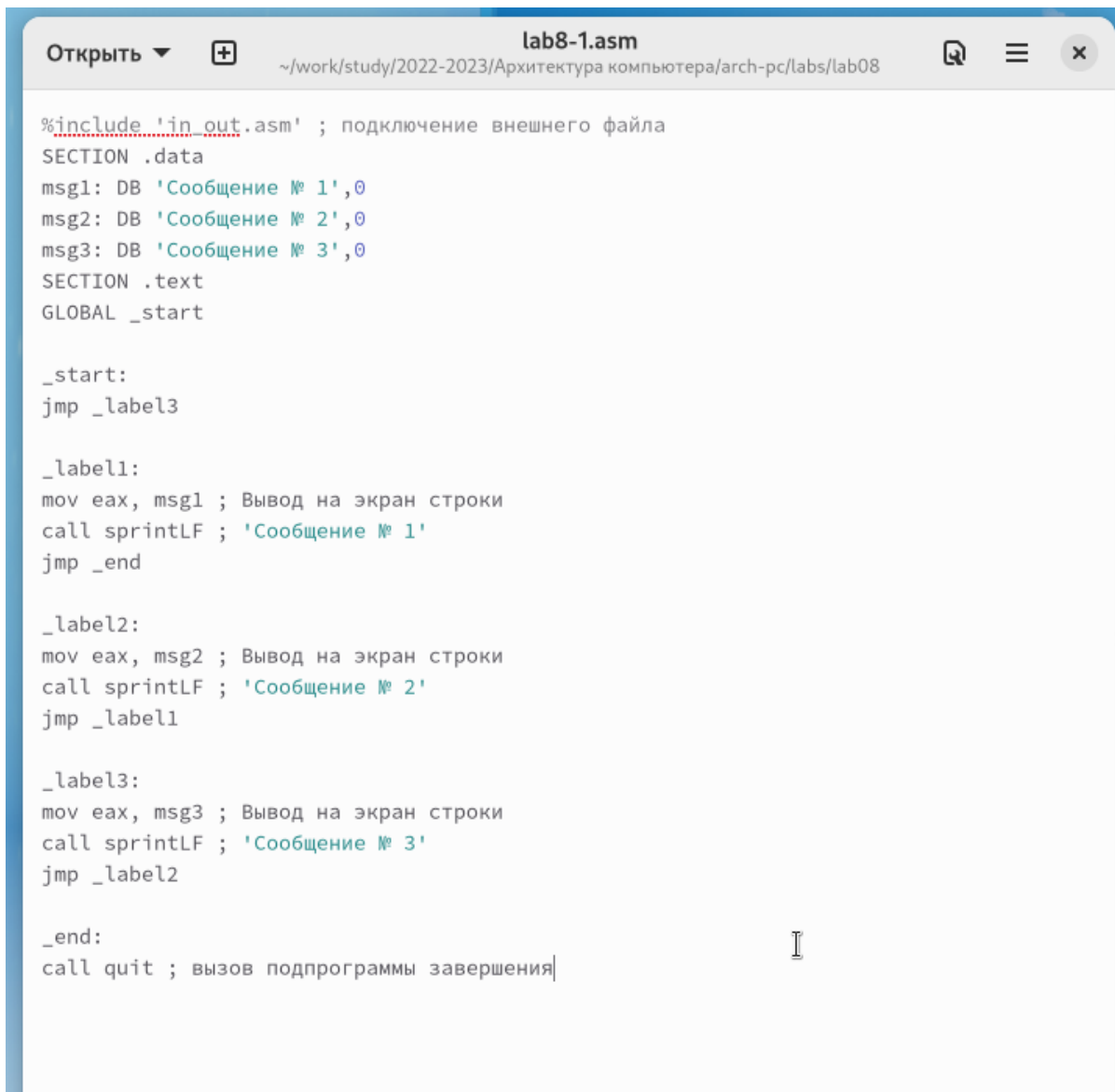


```
gislen@fedora:~/work/study/2022-2023/Архитектура компью...
[gislen@fedora lab08]$ nasm -f elf lab8-1.asm
[gislen@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[gislen@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[gislen@fedora lab08]$
[gislen@fedora lab08]$
[gislen@fedora lab08]$ nasm -f elf lab8-1.asm
[gislen@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[gislen@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1
[gislen@fedora lab08]$
```

Рис. 4.4: Программа lab8-1.asm:

Измените текст программы добавив или изменив инструкции `jmp`, чтобы вывод программы был следующим (рис. 4.5, 4.6):

```
user@dk4n31:~$ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
user@dk4n31:~$
```



```
Открыть ▾ + lab8-1.asm
~/work/study/2022-2023/Архитектура компьютера/arch-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

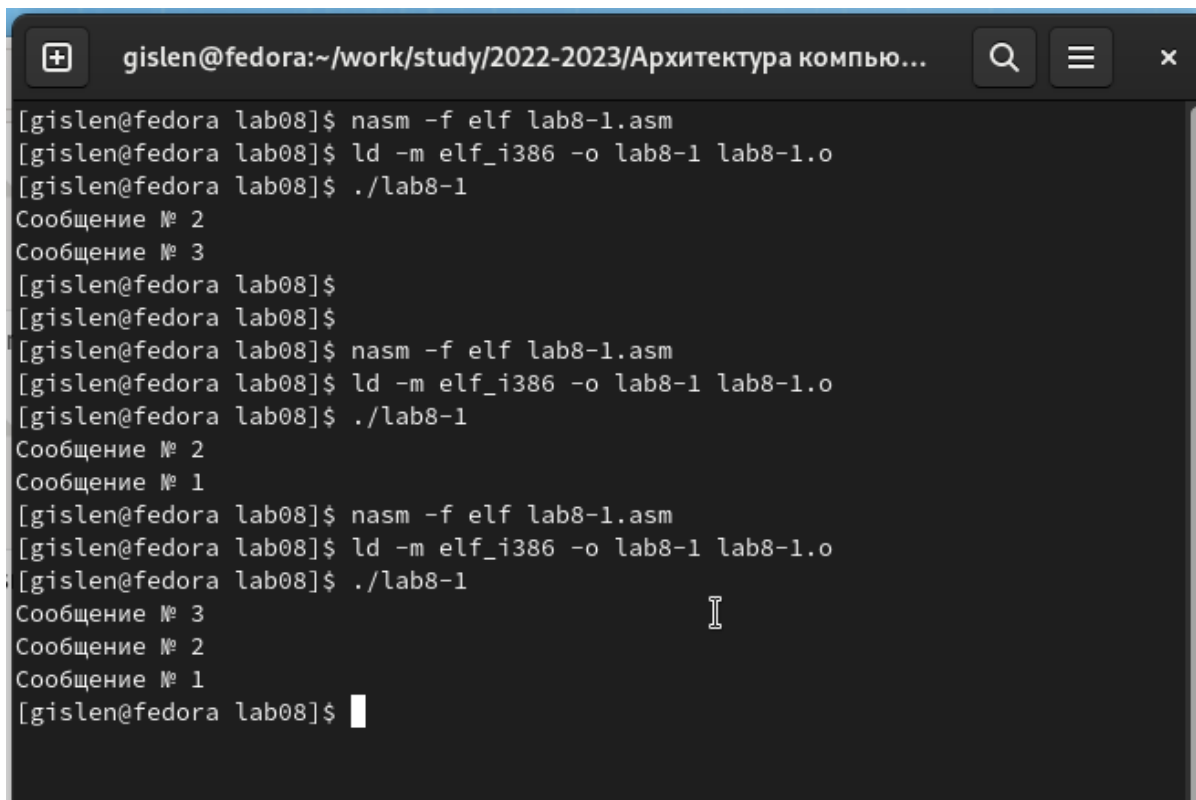
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
jmp _end

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

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

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.5: Файл lab8-1.asm



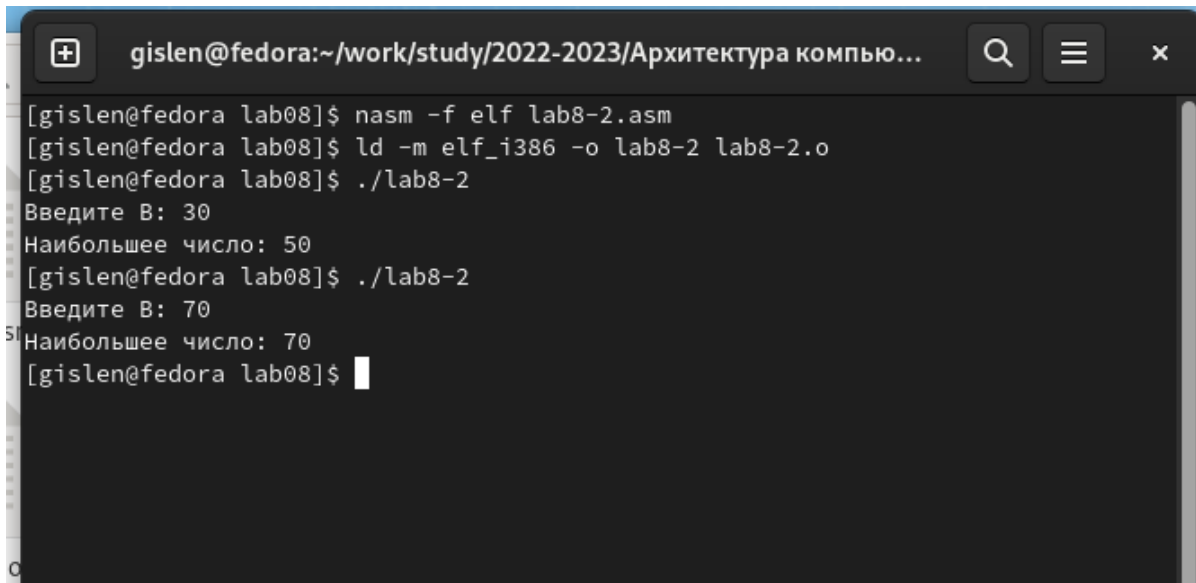
```
gislen@fedora:~/work/study/2022-2023/Архитектура компью...
[gislen@fedora lab08]$ nasm -f elf lab8-1.asm
[gislen@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[gislen@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[gislen@fedora lab08]$
[gislen@fedora lab08]$
[gislen@fedora lab08]$ nasm -f elf lab8-1.asm
[gislen@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[gislen@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1
[gislen@fedora lab08]$ nasm -f elf lab8-1.asm
[gislen@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[gislen@fedora lab08]$ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[gislen@fedora lab08]$
```

Рис. 4.6: 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: A, B и C. Значения для A и C задаются в программе, значение B вводится с клавиатуры. Создайте исполняемый файл и проверьте его работу для разных значений B. (рис. 4.7, 4.8)

```
%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:
; ----- 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'
```

Рис. 4.7: Файл lab8-2.asm



```
[gislen@fedora lab08]$ nasm -f elf lab8-2.asm
[gislen@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[gislen@fedora lab08]$ ./lab8-2
Введите B: 30
Наибольшее число: 50
[gislen@fedora lab08]$ ./lab8-2
Введите B: 70
Наибольшее число: 70
[gislen@fedora lab08]$
```

Рис. 4.8: 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. 4.9)


```
1      %include 'in out.asm'
2
3      <1> ;----- slen -----
4      <1> ; Функция вычисления длины сообщения
5      <1> slen:
6      <1>     push    ebx
7      <1>     mov     ebx, eax
8      <1> nextchar:
9      <1>     cmp     byte [eax], 0
10     <1>     jz      finished
11     <1>     inc     eax
12     <1>     jmp     nextchar
13
14     <1> finished:
15     <1>     sub     eax, ebx
16     <1>     pop     ebx
17     <1>     ret
18
19
20     <1> ;----- sprint -----
21     <1> ; Функция печати сообщения
22     <1> ; входные данные: mov eax, <message>
23     <1> sprint:
24     <1>     push    edx
25     <1>     push    ecx
26     <1>     push    ebx
27     <1>     push    eax
28     <1>     call   slen
29
30     <1>     mov     edx, eax
31     <1>     pop     eax
```

Рис. 4.9: Файл листинга lab8-2

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

строка 9

- 9 - номер строки
- 00000003 - адрес
- 803800 - машинный код
- `mp byte [eax], 0` - код программы

строка 10

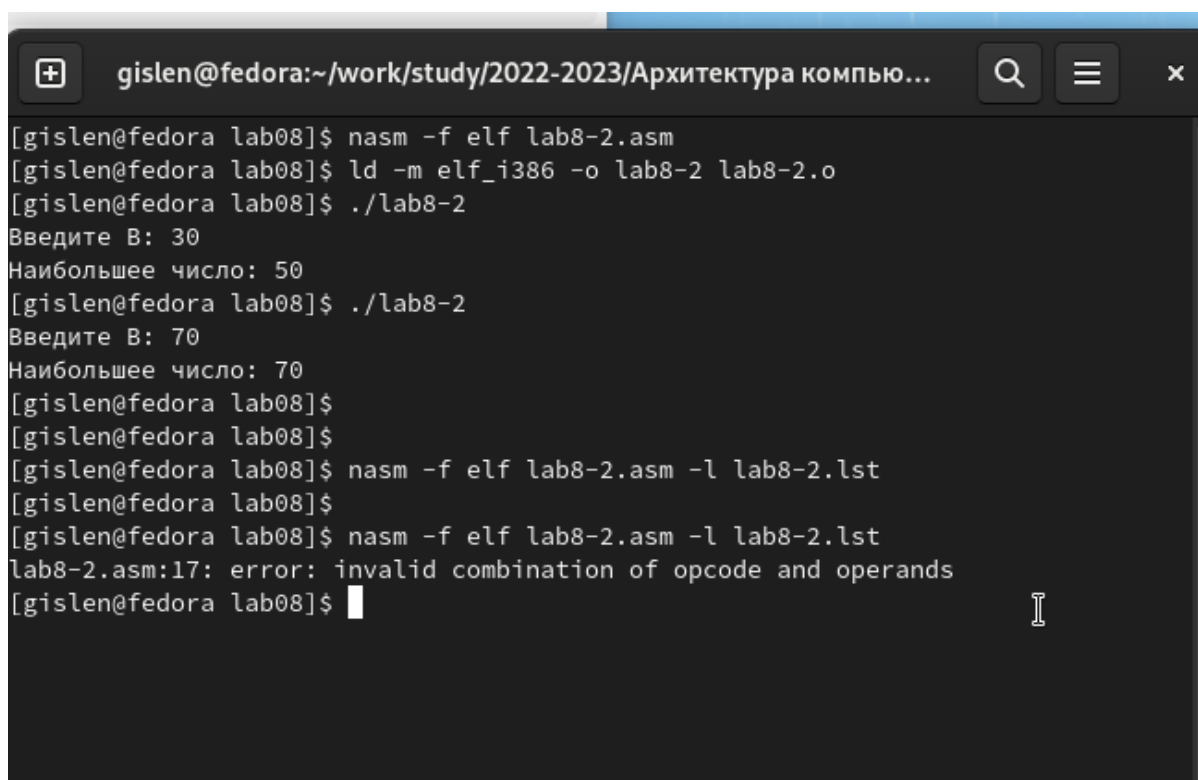
- 10 - номер строки
- 00000006 - адрес

- 7403 - машинный код
- jz finished - код программы

строка 11

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

Откройте файл с программой lab8-2.asm и в любой инструкции с двумя операндами удалить один операнд. Выполните трансляцию с получением файла листинга (рис. 4.10,4.11)



```

gislen@fedora:~/work/study/2022-2023/Архитектура компью...
[gislen@fedora lab08]$ nasm -f elf lab8-2.asm
[gislen@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[gislen@fedora lab08]$ ./lab8-2
Введите В: 30
Наибольшее число: 50
[gislen@fedora lab08]$ ./lab8-2
Введите В: 70
Наибольшее число: 70
[gislen@fedora lab08]$
[gislen@fedora lab08]$
[gislen@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-2.lst
[gislen@fedora lab08]$
[gislen@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-2.lst
lab8-2.asm:17: error: invalid combination of opcode and operands
[gislen@fedora lab08]$

```

Рис. 4.10: ошибка трансляции lab8-2

```

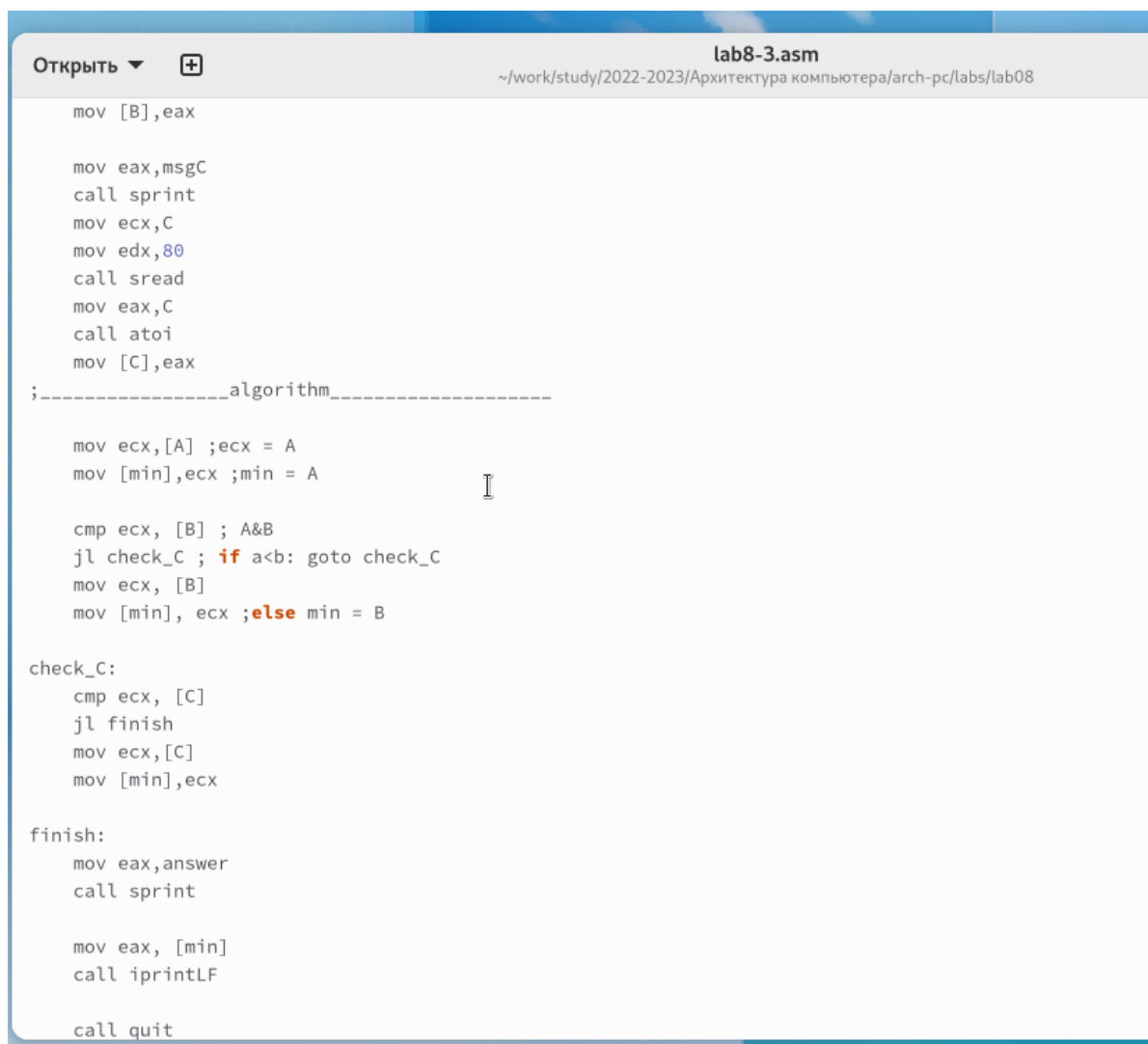
4 0000001C BED0BBD18CD188D0B5-
4 00000025 D0B520D187D0B8D181-
4 0000002E D0BBD0BE3A2000
5 00000035 32300000      A dd '20'
6 00000039 35300000      C dd '50'
7                          section .bss
8 00000000 <res Ah>      max resb 10
9 0000000A <res Ah>      B resb 10
10                          section .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                          mov ecx,
17                          error: invalid combination of opcode and operands
18 000000F2 BA0A000000      mov edx,10
19 000000F7 E847FFFFFF      call sread
20                          ; ----- Преобразование 'B' из символа в число
21 000000FC B8[0A000000]      mov eax,B
22 00000101 E896FFFFFF      call atoi ; Вызов подпрограммы перевода символа в число
23 00000106 A3[0A000000]      mov [B],eax ; запись преобразованного числа в 'B'
24                          ; ----- Записываем 'A' в переменную 'max'
25 0000010B 8B0D[35000000]      mov ecx,[A] ; 'ecx' = A
26 00000111 890D[00000000]      mov [max],ecx ; 'max' = A
27                          ; ----- Сравниваем 'A' и 'C' (как символы)
28 00000117 3B0D[39000000]      cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 0000011D 7F0C      jg check_B ; если 'A>C', то переход на метку 'check_B',
30 0000011F 8B0D[39000000]      mov ecx,[C] ; иначе 'ecx' = C
31 00000125 890D[00000000]      mov [max],ecx ; 'max' = C

```

Рис. 4.11: 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. 4.12,4.13)

для варианта 20 - 95,2,61



```
Открыть ▾ + lab8-3.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08

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

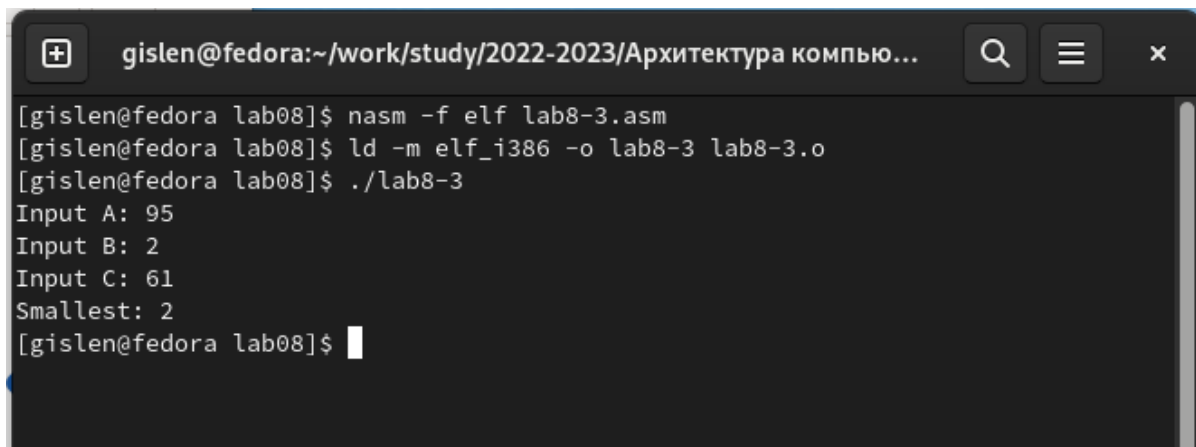
check_C:
cmp ecx, [C]
jl finish
mov ecx,[C]
mov [min],ecx

finish:
mov eax,answer
call sprint

mov eax, [min]
call iprintLF

call quit
```

Рис. 4.12: Файл lab8-3.asm



```
gislen@fedora:~/work/study/2022-2023/Архитектура компью...
[gislen@fedora lab08]$ nasm -f elf lab8-3.asm
[gislen@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[gislen@fedora lab08]$ ./lab8-3
Input A: 95
Input B: 2
Input C: 61
Smallest: 2
[gislen@fedora lab08]$
```

Рис. 4.13: 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. 4.14, 4.15)

для варианта 20

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

```
report.md

call atoi
mov [A],eax

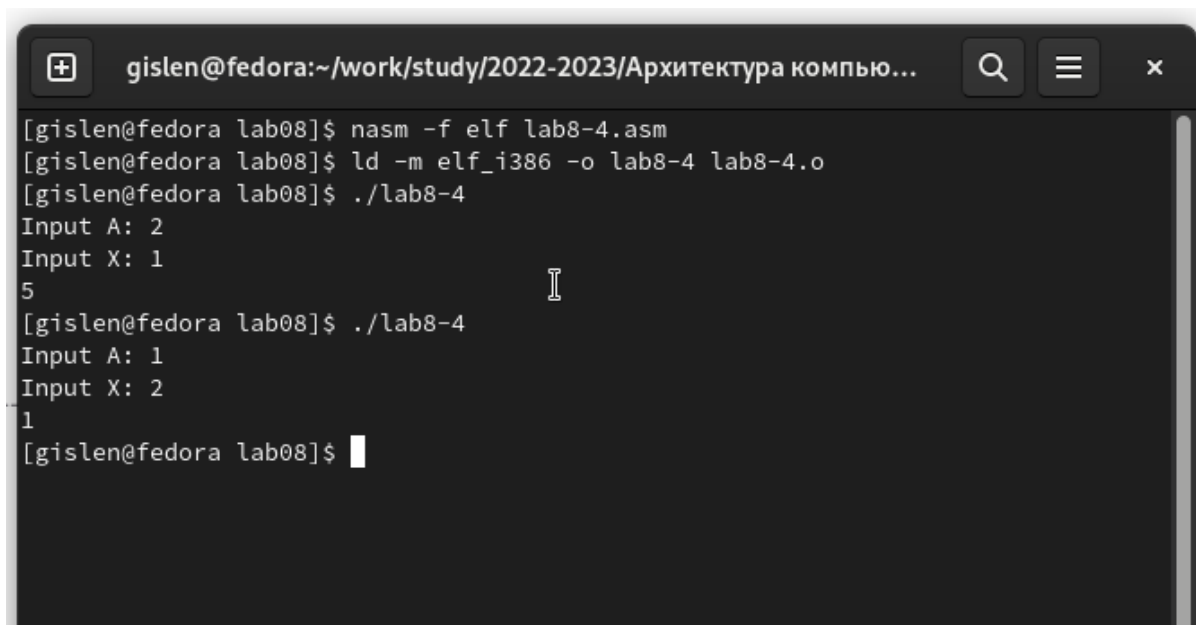
mov eax,msgX
call sprint
mov ecx,X
mov edx,80
call sread
mov eax,X
call atoi
mov [X],eax

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

mov ebx, [X]
mov edx, [A]
cmp ebx, 2
jae first
jmp second

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

Рис. 4.14: Файл lab8-4.asm



A terminal window with a dark background and light text. The title bar at the top reads "gislen@fedora:~/work/study/2022-2023/Архитектура компью...". The terminal content shows the following sequence of commands and output:

```
[gislen@fedora lab08]$ nasm -f elf lab8-4.asm
[gislen@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o
[gislen@fedora lab08]$ ./lab8-4
Input A: 2
Input X: 1
5
[gislen@fedora lab08]$ ./lab8-4
Input A: 1
Input X: 2
1
[gislen@fedora lab08]$
```

Рис. 4.15: Программа lab8-4.asm

5 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